ProgressXML

Version 1.3 / Revision 2024-01-28

Ein innovatives Datenübertragungskonzept für Fertigteilhersteller und Bewehrungsbetriebe

Herausgegeben von:
PROGRESS GROUP

In Abstimmung mit:
VeriCon Ingenieurs BV & VeriCon Solutions BV, Veldhoven (NL)
Precast Software Engineering GmbH, Salzburg (A)
Gesys GmbH & Co. KG, Kißlegg (D)
IDAT GmbH, Darmstadt (D)
Softbauware GmbH, Langen (D)

Index:

1 Allgemeines

1.1 Anwendungsbereich

Das Datenformat ***progressXML*** (im Folgenden auch **PXML** genannt) ist ein auf XML basierendes Datenformat für die Datenerstellung und Produktionssteuerung in Betonfertigteilwerken.

Im Einzelnen gibt es 2 unterschiedliche Anwendungsbereiche:

- Schnittstelle zwischen Systemen verschiedener Hersteller →*Standard-Tags*
- Interne (proprietäre) Datenspeicherung von CAD/CAM-Systemen→*Interne Tags*

Im vorliegenden Dokument werden die **Standard-Tags** beschrieben: das sind jene Felder des PXML-Formates, die verbindlich definiert sind und für den Datenaustausch zwischen verschiedenen Systemen verwendet werden können. Es ist *nicht* notwendig, alle Standard-Tags zu verwenden; man sollte jedoch versuchen, beim Datenaustausch ausschließlich die Standard-Tags zu verwenden.

Neben den Standard-Tags kann jeder Software-Hersteller für seine interne Datenrepräsentation beliebige proprietäre **interne Tags** definieren und verwenden. Diese sollen beim Datenaustausch zwischen verschiedenen Systemen einfach ignoriert werden.

Es wird empfohlen, internen Tags stets mit Präfix `I_` voranzustellen; dadurch werden Namens-Konflikte mit später eingeführten Standard-Tags vermieden (Standard-Tags beginnen niemals mit `I_`)[1].

Aus obigen Ausführungen ergeben sich für systemübergreifende PXML-Import-Module folgende Forderungen:

a) Es werden nur Standard-Tags gelesen.
b) Das Fehlen von Standard-Tags muss toleriert werden (Default-Werte werden angenommen, siehe Abschnitt 1.2).
c) Die PXML-Datei kann beliebig viele zusätzliche Tags enthalten (interne Tags), die beim Import einfach ignoriert werden.

1.2 Default-Werte

Beim Fehlen eines Tags werden Default-Werte angenommen. Die Default-Werte sind für jeden Datentyp einheitlich vorgegeben:

- **string**: ""
- **double**: 0
- **bool**: false

Bei anderen Datentypen, als den oben angeführten, sollten niemals Default-Werte angenommen werden. Das gilt insbesondere auch für Integer-Werte: da dies häufig IDs beinhalten, die auch 0 sein könnten, gibt es für Integers keine Default-Wert-Regelung.

[1] Eine Variante dieser Konvention ist es die Präfixe **I_P_** und **I_V_** zu verwenden: ersterer wird für *persistente* interne Tags verwendet, das sind jene, die auch auf File geschrieben werden. Letzterer wird für *flüchtige* (*volatile*) interne Tags verwendet, die *nicht* auf File geschrieben werden und i.allg. eine redundante Information enthalten (Cache-Felder).

1.3 Character Encoding

Es wird empfohlen in PXML-Dateien das UTF-8-Encoding zu verwenden. Man vermeidet dadurch viele Probleme und Komplikationen.

Da es sich aber um eine XML-Datei handelt, ist die Wahl des UTF-8-Codes jedoch nicht zwingend, sondern es gelten allgemein die Encoding-Regeln für XML-Dokumente:

a) Wenn es keine Encoding-Deklaration[2] gibt, und kein BOM (Byte Order Mark) vorhanden ist, handelt es sich um UTF-8.

b) Für UTF-16 und UTF-32 Encodings ist ein entsprechender BOM-Eintrag zwingend. Eine Encoding-Deklaration ist zwar nicht zwingend, aber empfohlen.

c) Falls ein altes (nicht-Unicode-fähiges) Encoding verwendet wird, muss eine explizite Encoding Deklaration angeführt werden. Typischerweise ist das "iso-8859-1" oder "windows-1252", für den üblichen sogenannten "ANSI"-Code.

1.4 Versioning

Es gibt 2 Versionskennungen: **Major-Version** und **Minor-Version**. Die Kombination *Major-Version /Minor-Version* stimmt mit der Versionskennung des vorliegenden Dokumentes überein.

Beachte: diese Versionskennungen beziehen sich nur auf die Standard-Tags; falls es ein Versioning für interne Tags gibt, muss dies über eine zusätzliche proprietäre Versionskennung erfolgen.

Major-Version (Schema-Version)

Die *Major-Version* wird im XML-Namespace festgelegt. Der Namespace hat folgendes Format:

`http://progress-m.com/ProgressXML/Version1`

Die *Major-Version* wird nur dann verändert, wenn sich die Schema-Syntax auf inkompatible Weise ändert.

Änderungen der *Major-Version* sollten, wenn möglich, vermieden werden. Ein Wechsel der *Major-Version* ist normalerweise nur dann nötig, wenn die vorhergehende Version (zumindest vorübergehend) weiterhin gepflegt werden muss.

Minor-Version

Die *Minor-Version* wird im DocInfo-Block eingetragen.

Die verschiedenen Minor-Versionen müssen syntaktisch kompatibel zueinander sein. D.h. eine Version kann Tags besitzen, die es in der anderen Version nicht gibt, oder ein- und derselbe Tag kann in verschiedenen Versionen eine unterschiedliche semantische Bedeutung haben; die Schema-Syntax muss jedoch kompatibel sein.

Beispiel 1: eine Winkelangabe, die ursprünglich in Grad war, wird in einer neuen Version als Zehntelgrad-Wert interpretiert.

Beispiel 2: eine Integer-Winkelangabe einer alten Version wird in einer neuen Version nicht mehr verwendet; stattdessen wird ein neues Tag mit einer Fließkomma-Winkel-Angabe eingeführt.

Beachte: Ergänzungen, die voll abwärtskompatibel sind, können auch innerhalb einer Minor-Version hinzugefügt werden. Typischerweise handelt es sich hierbei um Felder, die zusätzlich in den PXML-Standard aufgenommen wurden.

---

[2] Als **Encoding Declaration** ist hier eine Kopfzeile der Form
<?xml version="1.0" encoding="iso-8859-1" ?>
zu verstehen.

1.5 Kompatibilität mit UNICAM

Zielsetzung

Bei der Definition des ProgressXML-Formates wurde versucht eine weitgehende Kompatibilität mit der UNICAM CAD/CAM-Schnittstelle zu erreichen. Insbesondere soll es möglich sein, bei Konvertierungen

UNICAM → ProgressXML → UNICAM

wieder die ursprüngliche Datei zu erhalten (von wenigen Ausnahmen abgesehen).

Diese Forderung hat einige Elemente, Benennungen und Strukturen in die Spezifikation einfließen lassen, die an und für sich eher irrational erscheinen.

Wenn ohne Versions-Angabe auf die UNICAM-Dokumentation verwiesen wird, sind i. allg. die Versionen 5.2 oder 6.0 der UNICAM-Schnittstelle gemeint.

Nicht unterstützte UNICAM-Elemente

Einige Elemente der UNICAM CAD/CAM-Schnittstelle sind in PXML bewusst nicht integriert worden:

1) **Ist-Stückzahl:** wird beim Export nach UNICAM immer 0 gesetzt.
2) **Endhaken-Biegeformen:** es werden nur freie Biegeformen unterstützt; beim Import aus UNICAM werden Endhaken-Biegeformen in freie Biegeformen konvertiert.

1.6 PXML Delegate Files und PXML Include Files

Häufig sind an der Produktionsdatenerstellung 2 getrennte Systeme beteiligt: ein CAD-System und ein ERP-System. Hierbei liefert das CAD-System die geometrischen Informationen der Zeichnung. Das ERP-System liefert dagegen typischerweise Auftragskopfdaten, wie Auftragsnummer, Kundeninformationen und Liefertermin. Das **PXML Delegate File** trägt diesem Bedarf Rechnung, indem es ermöglicht, die ERP-Informationen mit den CAD-Informationen zu verschmelzen.

Das *PXML Delegate File* selber wird typischerweise von einem ERP-System generiert. Es enthält dementsprechend die Auftragskopfdaten, und bei Serienproduktionen eventuell auch Auftrags-Stückzahlen. Die geometrischen Detailinformationen des herzustellenden Produkts werden im *Delegate File* aber nicht direkt angegeben, sondern an ein **Include-File** "delegiert", das typischerweise vom CAD bereitgestellt wird.

Das *Delegate File* ist an sich ein ganz normales PXML-File. An Stelle der geometrischen Produkt-Details finden sich aber *Include*-Direktiven, welche *Include-Dateipfade* angeben[3]. Die *Include Files* (CAD-Dateien) sind auch wieder ganz normale PXML-Dateien[4]. Das folgende Beispiel zeigt, wie ein *Delegate File* mit 2 *Include Files* zu einem einzigen, vollständigen PXML-File verschmolzen werden[5]:

Delegate File \\SrvXY\Examples\delegate.pxml:

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
	<DocInfo>
		<MajorVersion>1</MajorVersion>
		<MinorVersion>3</MinorVersion>
	</DocInfo>
	<Order>
		<OrderNo>100000000041</OrderNo>
		<DeliveryDate>2013-11-05T10:50:07+01:00</DeliveryDate>
		<Product>
			<ElementNo></ElementNo>	<!--Ignore3-->
			<PieceCount>3</PieceCount>
			<Include>\\SrvXY\Examples\CADFiles\abcd1.pxml</Include>
		</Product>
		<Product>
			<PieceCount>5</PieceCount>
			<Include>CADFiles\abcd2.pxml</Include>
			<Slab>
				<X>600</X>
			</Slab>
		</Product>
	</Order>
</PXML_Document>
```

[3] Der Pfad zu einem *Include File* kann absolut oder relativ zum Verzeichnis angegeben werden, in dem sich das *Delegate File* befindet.

[4] Manchmal werden auch Dateien mit anderen Formaten inkludiert (z.B. UNICAM, BVBS); diese Dateien müssen hierbei dann aber während des Import-Vorgangs in PXML-Dateien gewandelt werden.

[5] Alle mit "Ignore" gekennzeichneten Angaben sollten gar nicht angegeben werden. Sie sind im Beispiel nur angeführt, um die Include-Regeln exakt definieren zu können.

CAD File \\SrvXY\Examples\CADFiles\abcd1.pxml:

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
	<DocInfo>
		<MajorVersion>1</MajorVersion>
		<MinorVersion>3</MinorVersion>
	</DocInfo>
	<Order>
		<Component>C1</Component>
		<Product>
			<ElementNo>E1</ElementNo>
			<ProductType>DW</ProductType>
			<Slab>
				<PartType>01</PartType>
			</Slab>
		</Product>
	</Order>
</PXML_Document>
```

CAD File \\SrvXY\Examples\CADFiles\abcd2.pxml:

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
	<DocInfo>
		<MajorVersion>1</MajorVersion>
		<MinorVersion>3</MinorVersion>
	</DocInfo>
	<Order>	<!--Ignore5-->
	</Order>
	<Order>
		<OrderNo>aa2</OrderNo>	<!--Ignore1-->
		<Component>C2</Component>	<!--Ignore6-->
		<Product>
			<ElementNo>E2</ElementNo>
			<ProductType>DW</ProductType>
			<PieceCount>1</PieceCount>	<!--Ignore4-->
			<Slab>
				<PartType>01</PartType>
			</Slab>
		</Product>
		<Product>	<!--Ignore2-->
			<ElementNo>E3</ElementNo>
		</Product>
	</Order>
	<Order>
		<Product>	<!--Ignore2-->
			<ElementNo>E5</ElementNo>
		</Product>
	</Order>
</PXML_Document>
```


Resulting merged File:

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
	<DocInfo>
		<MajorVersion>1</MajorVersion>
		<MinorVersion>3</MinorVersion>
	</DocInfo>
	<Order>
		<OrderNo>100000000041</OrderNo>
		<Component>C1</Component>
		<DeliveryDate>2013-11-05T10:50:07+01:00</DeliveryDate>
		<Product>
			<ElementNo>E1</ElementNo>
			<ProductType>DW</ProductType>
			<PieceCount>3</PieceCount>
			<Slab>
				<PartType>01</PartType>
			</Slab>
		</Product>
		<Product>
			<ElementNo>E2</ElementNo>
			<ProductType>DW</ProductType>
			<PieceCount>5</PieceCount>
			<Slab>
				<X>600</X>
			</Slab>
			<Slab>
				<PartType>01</PartType>
			</Slab>
		</Product>
	</Order>
</PXML_Document>
```

Die *Include*-Integration erfolgt im Einzelnen nachfolgenden Regeln:

a) Eine *Include*-Anweisung kann theoretisch auf jeder Hierarchiestufe stehen. Der bei weitem wichtigste Fall ist aber jener eines *Include* auf *Product*-Ebene, so wie im genannten Beispiel. Die Ebene, auf der das *Include* steht, sei fortan als **Include Level** bezeichnet.

b) Alles was im *Include File* hierarchisch oberhalb oder auf *Include Level* liegt, wird ignoriert, wenn es bereits gesetzt ist, entweder weil es im originalen *Delegate File* gesetzt war (siehe "Ignore1" im Beispiel), oder weil es durch ein bereits verarbeitetes *Include* gesetzt wurde (siehe "Ignore6" im Beispiel). Siehe "Ignore3" und "Ignore4" als weitere Beispiele. Leerstrings werden hierbei als Null-Werte angesehen.

c) Wenn das *Include File* auf *Include Level* mehrere Objekte hat, wird nur das erste passende Objekt genommen (siehe "Ignore2" im Beispiel).

Als Alternative zu den bisher genannten einfachen *Include* Direktiven kann man auch **komplexe Include Directiven** angeben:


```
<?xml version="1.0"?>
<Include>
	<FileName>\\SrvXY\CADFiles\abcd1.pxml</FileName>
	<Filter>
		<IncludeFilter>
			<DataTableName>Slab</DataTableName>
			<DataColumnName>PartType</DataColumnName>
			<Condition>\s*01\s*</Condition>	<!--RegEx condition-->
		</IncludeFilter>
		<IncludeFilter>
			<DataTableName>Steel</DataTableName>
			<DataColumnName>Name</DataColumnName>
			<Condition>\s*cageXy\s*</Condition>	<!--RegEx condition-->
		</IncludeFilter>
	</Filter>
</Include>
```

Diese *komplexe Include Direktive* muss wie folgt als Text-Feld eingebettet werden:

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
	<DocInfo>
		<MajorVersion>1</MajorVersion>
		<MinorVersion>3</MinorVersion>
	</DocInfo>
	<Order>
		<OrderNo>100000000041</OrderNo>
		<Product>
			<PieceCount>3</PieceCount>
			<Slab>
				<Steel>
					<Include>
&lt;?xml version="1.0"?>
&lt;Include&gt;
	&lt;FileName&gt;\\SrvXY\CADFiles\abcd1.pxml</FileName&gt;
	&lt;Filter&gt;
		&lt;IncludeFilter&gt;
			&lt;DataTableName&gt;Slab&lt;/DataTableName&gt;
			&lt;DataColumnName&gt;PartType</DataColumnName&gt;
			&lt;Condition&gt;\s*01\s*&lt;/Condition&gt;
		&lt;/IncludeFilter&gt;
		&lt;IncludeFilter&gt;
			&lt;DataTableName&gt;Steel&lt;/DataTableName&gt;
			&lt;DataColumnName&gt;Name&lt;/DataColumnName&gt;
			&lt;Condition&gt;\s*cageXy\s*&lt;/Condition&gt;
		&lt;/IncludeFilter&gt;
	&lt;/Filter&gt;
&lt;/Include&gt;
					</Include>
				</Steel>
			</Slab>
		</Product>
	</Order>
</PXML_Document>
```

In diesem Beispiel wird der erste Steel-Block inkludiert, der den Namen „cageXY“ hat und in einer ersten Wandhälfte liegt.

UNICAM Delegate File

Auch ein UNICAM-File kann als *Delegate*-File interpretiert werden (ab Version 6.0).

Hierbei werden die SLABDATE Info-Felder 1 bis 4 verwendet:

Info1: enthält die Delegate-File-Kennung, und zwar den festen Text „#include“

Info2+Info3+Info4: enthält den Pfad zur PXML-Datei (der Pfad wird durch zusammenbinden dieser 3 Felder gebildet, wobei bei jedem dieser Felder die Leerzeichen am Ende weggeschnitten werden).

Die Pfadangabe kann hierbei folgende Makros enthalten:

<A>: Auftragsname (aus Zeile 3 des HEADER)

<E>: Element-Name

<P>: Projekt (Zeile 4 des HEADER)

<K>: Kunde (Zeile 12 des HEADER)

Ein Beispiel für die Info-Zeilen wäre wie folgt:

Info1: **#include**

Info2: **\\srvXY\CAD**

Info3: **<K>\<P>**

Info4: **<A>_<E>.PXML**

Das UNICAM *Delegate*-File kann auch einen „Drehwinkel des Elements“ beinhalten (SLABDATE, Zeile 4, Position 55-57 – bei Version 6.0). Diese Drehung kann nun bereits im CAD durchgeführt worden sein, oder sie kann nachträglich vom Leitsystem vorgegeben werden. Deshalb muss dieser Winkel, abzüglich des Winkels aus *Product.RotationPosition*, als Drehvorgabe verwendet werden. D.h. beim Erstellen der zusammengeführten Datei muss das Element um den Winkel der genannte Winkeldifferenz gedreht werden. Der Dreh-Wert aus der UNICAM-*Delegate*-Datei ist dann als neuer Wert für *Product.RotationPosition* zu übernehmen.

2 Struktur-Übersicht

Die folgende Darstellung beschreibt die Struktur des PXML-Files. Details zu den einzelnen Feldern sind weiter unten angeführt.

Die rechts angeführten roten Zahlenangaben legen fest, wie oft ein Eintrag vorkommen kann oder muss:

- **1** = genau einmal
- **0,1** = 0 oder 1 mal
- **>=1** = mindestens einmal
- **n** = beliebig oft (auch 0 mal)

Die Längen der Zeichenketten sind grundsätzlich beliebig. Beim Export nach UNICAM werden die Zeichenketten jedoch auf eine maximale Länge reduziert, bzw. auf eine bestimmte Länge mit Leerzeichen aufgefüllt.

Die angegebenen Texte (z.B. aaaaaaaa) sind Beispiele. Wo nicht anders erwähnt, sind alle Texte beliebige alphanumerische Zeichenfolgen; es ist aber zu beachten, dass einige dieser freien Texte beim UNICAM-Export zu numerischen Werten konvertiert werden müssen. Jene Werte, die bereits in der PXML-Definition numerisch sind, haben in der Tabelle die Anmerkung **Int**, **Bool**, oder **Double**.

<PXML Document>		**1**
<DocInfo GlobalID="aaa">		**1**
<MajorVersion>1</MajorVersion>	Int	1
<MinorVersion>3</MinorVersion>	Int	1
<Comment>aaaaaaaaaaaaa</Comment>		0,1
<ConvertConventions>aaa#bbb#ccc</ConvertConventions>		0,1
<Mode>		**n**
<ID>aaaaa</ID>		1
<Val>true</Val>		0,1
</Mode>		
</DocInfo>		
<Order GlobalID="aaa">		**n**
<OrderNo>aaaaaaaa</OrderNo>		0,1
<Structure>cccc</Structure>		0,1
<Building>cccc</Building>		0,1
<Storey>cccc</Storey>		0,1
<SubStorey>cccc</SubStorey>		0,1
<Component>bbbbb</Component>		0,1
<DrawingNo>dddddd</DrawingNo>		0,1
<DrawingDate>dd.mm.yyyy</DrawingDate>		0,1
<DrawingRevision>ee</DrawingRevision>		0,1
<DrawingAuthor>aaaaaaaaaaaaaaaaaaaa</DrawingAuthor>		0,1
<ErpProjectUnit>dddddd</ErpProjectUnit>		0,1
<DeliveryDate>2010-04-16T11:46:48.933+02:00</DeliveryDate>		0,1
<GenericOrderInfo01>aaaaaaaaaaaaaaaaaaaa</GenericOrderInfo01>		0,1
: : : :		0,1
<GenericOrderInfo20>aaaaaaaaaaaaaaaaaaaa</GenericOrderInfo20>		0,1
<Comment>aaaaaaaaaaaaaaaaaaaa</Comment>		0,1
<OrderArea>1234.5</OrderArea>	Double	0,1
<ImportSource>aaaaaaaaaaaaaaaaaaaa</ImportSource>		0,1

<ImportSourceType>aaaaaaaaaaaaaaaaaaaa</ImportSourceType>		0,1
<ApplicationName>aaaaaaaaaaaaaaaaaaaa</ApplicationName>		0,1
<ApplicationGUID>aaaaaaaaaaaaaaaaaaaa</ApplicationGUID>		0,1
<ApplicationVersion>aaaaaaaaaaaaaaaaaaaa</ApplicationVersion>		0,1
<OrderInfo Type="AccountingPosition" GlobalID="aaa">		**n**
<Code>aaaaa</Code>		0,1
<OrderInfoVal Type="aa" V="bb" U="mm" Culture="en"/>		**n**
</OrderInfo>		
<Product GlobalID="aaa">		**n**
<ElementNo>111</ElementNo>		0,1
<ProductType>11</ProductType>		0,1
<TotalThickness>6666</TotalThickness>	Double	0,1
<DoubleWallsGap>777</DoubleWallsGap>	Double	0,1
<PieceCount>1111</PieceCount>	Int	0,1
<TurnWidth>3500</TurnWidth>	Double	0,1
<Comment>aaaaaaaaaa</Comment>		0,1
<RotationPosition>111</RotationPosition>	Double	0,1
<StackNo>111</StackNo>		0,1
<StackID>111</StackID>		0,1
<StackingSequence>111</StackingSequence>		0,1
<StackingLevel>111</StackingLevel>		0,1
<StackingX>11111</StackingX>	Double	0,1
<StackingY>11111</StackingY>	Double	0,1
<StackingZ>11111</StackingZ>	Double	0,1
<StackingAngle>111</StackingAngle>	Double	0,1
<StackingRotY>111</StackingRotY>	Double	0,1
<StackingRotX>111</StackingRotX>	Double	0,1
<P1X>111111</P1X>	Double	0,1
<P1Y>111111</P1Y>	Double	0,1
<P1Z>111111</P1Z>	Double	0,1
<P2X>111111</P2X>	Double	0,1
<P2Y>111111</P2Y>	Double	0,1
<P2Z>111111</P2Z>	Double	0,1
<P3X>111111</P3X>	Double	0,1
<P3Y>111111</P3Y>	Double	0,1
<P3Z>111111</P3Z>	Double	0,1
<AdditionInfo>aaaa</AdditionInfo>		0,1
<UnloadingInfo>aaaa</UnloadingInfo>		0,1
<TransportInfo>aaaa</TransportInfo>		0,1
<ItemPosition>aaaa@bbbb@cccc@dddd</ItemPosition>		0,1
<ElementInfo Type="AccArea" Inventory=true GlobalID="aaa">		**n**
<Code>aaaaaaa</Code>		0,1
<Description>aaaaaaa</Description>		0,1
<ObjectID>aaaaaaa</ObjectID>		0,1
<PieceCount>22</PieceCount>	Int	0,1
<Val1>123.4</Val1>	Double	0,1
<Val2>-987.2</Val2>	Double	0,1
<Unit>m³</Unit>		0,1
<Details>aaaaaaa</Details>		0,1

	<ElemInfoVal Type="Width" V="322" U="mm"/>		**n**
	</ElementInfo>		
	<Slab GlobalID="aaa">		**n**
(Legacy)	<SlabNo>111</SlabNo>		0,1
	<PartType>11</PartType>		0,1
(Legacy)	<ProductAddition>44</ProductAddition>		0,1
	<ProductionWay>aa</ProductionWay>		0,1
(Legacy)	<NumberOfMeansOfTransport>666</NumberOfMeansOfTransport>		0,1
(Legacy)	<TransportSequence>777</TransportSequence>		0,1
(Legacy)	<PileLevel>888</PileLevel>		0,1
(Legacy)	<TypeOfUnloading>99</TypeOfUnloading>		0,1
(Legacy)	<MeansOfTransport>00</MeansOfTransport>		0,1
	<ExpositionClass>aaaaaaa</ExpositionClass>		0,1
	<SlabArea>11.111</SlabArea>	Double	0,1
	<SlabWeight>55555.5</SlabWeight>	Double	0,1
	<ProductionThickness>2222</ProductionThickness>	Double	0,1
	<MaxLength>11111</MaxLength>	Double	0,1
	<MaxWidth>22222</MaxWidth>	Double	0,1
	<IronProjectionLeft>±3333</IronProjectionLeft>	Double	0,1
	<IronProjectionRight>±4444</IronProjectionRight>	Double	0,1
	<IronProjectionBottom>±5555</IronProjectionBottom>	Double	0,1
	<IronProjectionTop>±6666</IronProjectionTop>	Double	0,1
	<X>111111</X>	Double	0,1
	<Y>222222</Y>	Double	0,1
	<Z>222222</Z>	Double	0,1
	<RotX>111111</RotX>	Double	0,1
	<RotY>222222</RotY>	Double	0,1
	<RotZ>222222</RotZ>	Double	0,1
	<ProdX>111111</ProdX>	Double	0,1
	<ProdY>222222</ProdY>	Double	0,1
	<ProdZ>222222</ProdZ>	Double	0,1
	<ProdRotX>111111</ProdRotX>	Double	0,1
	<ProdRotY>222222</ProdRotY>	Double	0,1
	<ProdRotZ>222222</ProdRotZ>	Double	0,1
(Legacy)	<OrderPosition>aaaaaaaa</OrderPosition>		0,1
(Legacy)	<ProductGroup>bbbb</ProductGroup>		0,1
(Legacy)	<SlabType>33</SlabType>		0,1
(Legacy)	<ItemDesignation>c...c</ItemDesignation>		0,1
(Legacy)	<ProjectCoordinates>111111 222222…</ProjectCoordinates>		0,1
(Legacy)	<PositionInPileX>111111</PositionInPileX>	Double	0,1
(Legacy)	<PositionInPileY>111111</PositionInPileY>	Double	0,1
(Legacy)	<PositionInPileZ>111111</PositionInPileZ>	Double	0,1
(Legacy)	<AngleInPile>111111</AngleInPile>	Double	0,1
	<GenericInfo01>aaaaaaa…aaaaaa</GenericInfo01>		0,1
	<GenericInfo02>aaaaaaa…aaaaaa</GenericInfo02>		0,1
	<GenericInfo03>aaaaaaa…aaaaaa</GenericInfo03>		0,1
	<GenericInfo04>aaaaaaa…aaaaaa</GenericInfo04>		0,1
	<ReforcemInfo></ReforcemInfo>		0,1
	<Outline Type="lot" GlobalID="aaa">		**n**

<X>22222</X>	Double	0,1
<Y>22222</Y>	Double	0,1
<Z>22222</Z>	Double	0,1
<RotX>22222</RotX>	Double	0,1
<RotY>22222</RotY>	Double	0,1
<RotZ>22222</RotZ>	Double	0,1
<Height>22222</Height>	Double	0,1
<Name>bbbb...bbbb</Name>		0,1
<GenericInfo01>bbbb...bbbb</GenericInfo01>		0,1
<GenericInfo02>bbbb...bbbb</GenericInfo02>		0,1
<MountingInstruction>2</MountingInstruction>		0,1
<MountPartType>33</MountPartType>		0,1
<MountPartArticle>aaaaaaaaaa</MountPartArticle>		0,1
<MountPartIronProjection>333</MountPartIronProjection>	Double	0,1
<MountPartDirection>±555</MountPartDirection>	Double	0,1
<MountPartLength>66666</MountPartLength>	Double	0,1
<MountPartWidth>77777</MountPartWidth>	Double	0,1
<ConcretingMode>aa</ConcretingMode>		0,1
<ConcreteQuality>aaaaaaaa</ConcreteQuality>		0,1
<UnitWeight>4.444</UnitWeight>	Double	0,1
<Volume>33.333</Volume>	Double	0,1
<Layer>aaa</Layer>		0,1
<ObjectID>aaaaaaa</ObjectID>		0,1
<Shape GlobalID="aaa">		**n**
<Cutout>false</Cutout>	Bool	0,1
<RefHeight>33.333</RefHeight>	Double	0,1
<SVertex GlobalID="aaa">		**n**
<X>11111</X>	Double	0,1
<Y>22222</Y>	Double	0,1
<Bulge>33333</Bulge>	Double	0,1
<LineAttribute>33333</LineAttribute>	Double	0,1
<Profile>-10 10|0 0</Profile>		0,1
<DX>22222</DX>	Double	0,1
<DY>22222</DY>	Double	0,1
</SVertex>		
</Shape>		
</Outline>		
<Steel Type="mesh" GlobalID="aaa">		**n**
<X>22222</X>	Double	0,1
<Y>22222</Y>	Double	0,1
<Z>22222</Z>	Double	0,1
<RotX>22222</RotX>	Double	0,1
<RotY>22222</RotY>	Double	0,1
<RotZ>22222</RotZ>	Double	0,1
<ToTurn>true</ToTurn>	Bool	0,1
<StopOnTurningSide>true</StopOnTurningSide>	Bool	0,1
<Name>2</Name>		0,1
<GenericInfo01>aaa…aaa</GenericInfo01>		0,1
: : : :		0,1

<GenericInfo06>aaa…aaa</GenericInfo06>		0,1
<MeshType>2</MeshType>		0,1
<WeldingDensity>100</WeldingDensity>	Int	0,1
<BorderStrength>2</BorderStrength>	Int	0,1
<ProdX>±123</ProdX>	Double	0,1
<ProdY>±123</ProdY>	Double	0,1
<ProdZ>±123</ProdZ>	Double	0,1
<ProdRotX>±123</ProdRotX>	Double	0,1
<ProdRotY>±123</ProdRotY>	Double	0,1
<ProdRotZ>±123</ProdRotZ>	Double	0,1
<Layer>aaa</Layer>		0,1
<ObjectID>aaaaaaa</ObjectID>		0,1
<Bar GlobalID="aaa">		**n**
<ShapeMode>realistic</ShapeMode>		0,1
<ReinforcementType>3</ReinforcementType>		0,1
<SteelQuality>aaa</SteelQuality>		0,1
<PieceCount>55555</PieceCount>	Int	0,1
<Diameter>666</Diameter>	Double	0,1
<X>±88888</X>	Double	0,1
<Y>±99999</Y>	Double	0,1
<Z>±77777</Z>	Double	0,1
<RotZ>±123</RotZ>	Double	0,1
<ArticleNo>bbbbbbbbbb</ArticleNo>		0,1
<NoAutoProd>false</NoAutoProd>	Bool	0,1
<ExtIronWeight>444.444</ExtIronWeight>	Double	0,1
<Bin>123</Bin>		0,1
<Pos>aaa</Pos>		0,1
<Note>aaa</Note>		0,1
<Machine>aaa</Machine>		0,1
<BendingDevice>1</BendingDevice>		0,1
<Spacer GlobalID="aaa">		**n**
<Type>222</Type>	Int	0,1
<Position>33333</Position>	Double	0,1
</Spacer>		
<WeldingPoint GlobalID="aaa">		**n**
<WeldingOutput>77</WeldingOutput>	Double	0,1
<Position>33333</Position>	Double	0,1
<WeldingPointType>111</WeldingPointType>	Int	0,1
<WeldingPrgNo>222</WeldingPrgNo>	Int	0,1
<GroupID>aaa</GroupID>		0,1
</WeldingPoint>		
****		**n**
<RotX>±123</RotX>	Double	0,1
<BendY>±123</BendY>	Double	0,1
<L>33333</L>	Double	0,1
<R>22</R>	Double	0,1
****		
</Bar>		
<Girder GlobalID="aaa">		**n**

<PieceCount>55555</PieceCount>	Int	0,1
<X>±88888</X>	Double	0,1
<Y>±99999</Y>	Double	0,1
<Z>±77777</Z>	Double	0,1
<GirderName>aaaaaaaaaa</GirderName>		0,1
<Length>55555</Length>	Double	0,1
<AngleToX>±999</AngleToX>	Double	0,1
<NoAutoProd>true</NoAutoProd>	Bool	0,1
<Height>222</Height>	Double	0,1
<TopExcess>222</TopExcess>	Double	0,1
<BottomExcess>222</BottomExcess>	Double	0,1
<Weight>33.333</Weight>	Double	0,1
<TopFlangeDiameter>44</TopFlangeDiameter>	Double	0,1
<BottomFlangeDiameter>44</BottomFlangeDiameter>	Double	0,1
<DiagonalWireDiameter>44</DiagonalWireDiameter>	Double	0,1
<GirderType>2</GirderType>	Int	0,1
<MountingType>1</MountingType>	Int	0,1
<ArticleNo>aaa</ArticleNo>		0,1
<Machine>aaa</Machine>		0,1
<Period>000</Period>	Double	0,1
<PeriodOffset>111</PeriodOffset>	Double	0,1
<Width>80</Width>	Double	0,1
<AnchorBar GlobalID="aaa">		**n**
<Type>222</Type>	Int	0,1
<Length>111</Length>	Double	0,1
<Position>33333</Position>	Double	0,1
</AnchorBar>		
<GirderExt Type="SplicePos" GlobalID="aaa">		**n**
<Position>33333</Position>	Double	0,1
<Flags>0</Flags>	Int	0,1
<Val0>33333</Val0>	Double	0,1
<Val1>33333</Val1>	Double	0,1
<Val2>33333</Val2>	Double	0,1
<Val3>33333</Val3>	Double	0,1
</GirderExt>		
<Section GlobalID="aaa">		**n**
<L>195</L>	Double	0,1
<S>-100</S>	Double	0,1
<F>50</F>	Double	0,1
</Section>		
</Girder>		
<Alloc Type="Bar" GlobalID="aaa">		**n**
<GuidingBar>2</GuidingBar>	Int	0,1
<Region GlobalID="aaa">		**n**
<IntervalCount>5</IntervalCount>	Int	0,1
<Pitch>111</Pitch>	Double	0,1
<IncludeBegin>true</IncludeBegin>	Bool	0,1
<IncludeEnd>true</IncludeEnd>	Bool	0,1
<RefIndex>4</RefIndex>	Int	0,1

</Region>		
</Alloc>		
<SteelExt Type="Xyz" GlobalID="aaa">		**n**
<Info>aaaaaaaaaa</Info>		0,1
</SteelExt>		
</Steel>		
</Slab>		
</Product>		
</Order>		
<Feedback ItemType="Bar" GlobalID="aaa">		**n**
<MessageType>error</MessageType>		0,1
<Code>123ABC</Code>		0,1
<InfoValue>123XYZ</InfoValue>		0,1
<PieceCount>3</PieceCount>	Int	0,1
<MaterialType>16A</MaterialType>		0,1
<MaterialBatch>12345@AR177228C</MaterialBatch>		0,1
<MaterialWeight>12345</MaterialWeight>	Double	0,1
<ProdDate>2010-07-30T09:06:05+02:00</ProdDate>		0,1
<Machine>aaa</Machine>		0,1
<Description Culture="en" Text="aaaaaaaaaa"/>		n
<FbVal T="OrdNo" V="AA00048386"/>		**n**
</Feedback>		
</PXML_Document>		

3 Detail-Spezifikationen

3.1 Global ID

Jede PXML-Tabelle kann ein Attribut namens **GlobalID** haben, das eine globale Identifikation des entsprechenden Items ermöglicht. "Global" ist hierbei als systemübergreifend zu verstehen, im Gegensatz also, zu systeminternen IDs, die nur innerhalb eines Subsystems bekannt sind.

Die *GlobalID* wird typischerweise vom datenerzeugenden System vergeben (also im CAD oder im Leitrechner), und von den Subsystemen unverändert übernommen, um dann in Rückmeldungen an das übergeordnete System verwendet zu werden.

Manche Systeme verwenden eine numerische *GlobalID*, andere verwenden einen String (z.B. einen GUID-String). Wie für fast alle PXML-Felder, gilt aber auch für die *GlobalID*, dass ihre Verwendung optional ist; so ist es durchaus möglich, ganz ohne *GlobalID* zu arbeiten, bzw. die *GlobalID* nur in einigen PXML-Tabellen zu verwenden.

3.1.1 Eindeutigkeit der GlobalID

Die *GlobalID* soll innerhalb eines Item-Typs eindeutig sein, d.h. beispielsweise, zwei unterschiedliche *Bar*-Items sollen unterschiedliche *GlobalID*s haben (das betrifft nicht nur 2 *Bar*-Items, die zum selben *Steel*-Block gehören, sondern auch *Bar*-Items aus unterschiedlichen *Steel*-Blocks sollen unterschiedliche *GlobalID* haben). Die *GlobalID*s verschiedener Item-Typen (z.B. *Bar* und *Girder*) können dagegen Überschneidungen haben.

Die Eindeutigkeit der *GlobalID* ist aber keine feste PXML-Vorschrift, sondern lediglich ein Erfordernis der Anwendung, das – je nach konkreter Applikation – unterschiedlich ausgeprägt sein kann. So reicht es für eine *PTS*-Anfrage beispielsweise aus, wenn die *GlobalID*s innerhalb eines Anfragedokuments eindeutig sind. Für Produktionsrückmeldungen von Maschinen wird man dagegen typischerweise eine universellere Eindeutigkeit der *GlobalID*s benötigen. (*PTS*-Anfragen und Produktionsrückmeldungen werden weiter unten näher erläutert).

3.1.2 Sonderfall: GlobalID der DocInfo-Tabelle

Für die *DocInfo*-Tabelle hat die *GlobalID* eine etwas abgewandelte Bedeutung: sie identifiziert dort nicht den Tabelleneintrag (denn es gibt höchstens einen *DocInfo*-Tabelleneintrag), sondern das gesamte Dokument.

3.1.3 Automatisches Generieren von GlobalIDs

Falls vom datenerzeugenden System keine *GlobalID* übergeben wird, kann das Subsystem diese Identifikations-Kennung eigenständig generieren, um dann gezielte Rückmeldungen machen zu können. Es soll dabei folgendes Muster verwendet werden:

ParentItemGlobalID.ItemIndex

Hierbei ist "ParentItemGlobalID" die *GlobalID* des unmittelbar darüber liegenden Parent-Items, und "ItemIndex" ist der nullbasierte Element-Index des betreffenden Items innerhalb seiner unmittelbaren Parent-Umgebung. Da *Order*-Items keinen Parent haben, wird dort nur der ItemIndex genommen (ohne Präfix).

<u>Beispiel 1</u>: Ein *Bar*-Item habe die *GlobalID* "ABC"; für die Segmente dieses Eisens können dann die *GlobalID*s wie folgt generiert werden:

"ABC.0"
"ABC.1"
"ABC.2"
usw.

Beispiel 2: Alle PXML-Items seien ohne *GlobalID*. für das fünfte Eisen im dritten *Order*-Block kann man dann folgende *GlobalID* automatisch generieren:
"2.0.0.0.4"
(*Order*-Index = 2, *Product*-Index = 0; *Slab*-Index = 0, *Steel*-Index = 0; *Bar*-Index = 4).

3.1.4 Spätes generieren von GlobalIDs

Dort wo eine systemübergreifende Identifikation von Items notwendig ist, sind *GlobalIDs* hilfreich oder sogar notwendig. In anderen Fällen können solche IDs aber hinderlich sein, weil sie eine künstliche Unterscheidung einführen, wo das nicht notwendig wäre: Wenn zwei PXML-Objekte inhaltlich identisch sind, kann es softwaretechnisch effizient sein, diese über ein gemeinsames Objekt zu behandeln; diese Optimierung würde aber durch unterschiedliche *GlobalIDs* vereitelt. Falls die Unterscheidung der beiden Objekte aus Anwendungssicht notwendig ist, macht es auch Sinn, unterschiedliche *GlobalIDs* zu haben. Wenn aber aus Anwendungssicht keine Unterscheidung erforderlich ist, würden die *GlobalIDs* in künstlicher und störender Weise eine Unterscheidbarkeit einführen.

Es ist daher sinnvoll, *GlobalIDs* im Gesamtprozess immer erst dann zu generieren, wenn sie effektiv benötigt werden.

3.2 DocInfo

Der *DocInfo*-Block muss genau einmal vorhanden sein.

3.2.1 GlobalID

Der *GlobalID*-Eintrag ist optional und dient der Identifikation des gesamten Dokumentes, insbesondere in Hinblick auf *PTS*-Rückmeldungen.

3.2.2 Document Version

Die *MajorVersion* und die *MinorVersion* bezeichnen die Hauptversion der zugrundeliegenden PXML-Spezifikation. Siehe hierzu Abschnitt 1.4.

3.2.3 Comment

Im *Comment*-Feld kann ein beliebiger Kommentar zum Dokument eingetragen werden. Die *Comment*-Angabe ist optional, sollte aber vor allem bei *PTS*-Rückmeldungen gesetzt werden, um den Zustand des sendenden Systems zu beschreiben (Identifikation des Systems, Version, Parameter-Version).

3.2.4 ConvertConventions

Wenn eine PXML-Datei aus irgendwelchen Gründen nicht PXML-konform ist, kann man das in den *ConvertConventions* vermerken. Beim Importieren einer solchen Datei können die Daten dann geeignet konvertiert werden, um einen PXML-konformen Datenbestand zu erhalten.

Falls mehrere Konventionen angegeben werden sollen, sind diese durch ein #-Zeichen zu trennen.

PXML-basierte Systeme arbeiten intern ohne Berücksichtigung der *ConvertConventions*. Die *ConvertConventions* werden sofort beim Importieren der Daten abgearbeitet und dann gelöscht.

Es gibt grundsätzlich keine *ConvertConventions*, die zwingend akzeptiert werden müssen. Folgende Konventionen werden aber häufig akzeptiert:

- **SegmentsHaveOuterLen**: Die Segment.L-Angaben geben die Außenlänge des Segmentes an, und nicht (wie in PXML gefordert) die Kernmaßlänge.

3.2.5 Mode

Über *Mode*-Einträge können Begleitinformationen angegeben werden, die typischerweise Verarbeitungsanweisungen für den Datenempfänger enthalten.

Jeder *Mode*-Eintrag besteht aus einem *ID*-Feld, das die Bedeutung des Eintrags festlegt, und einem *Val*-Feld, das den Wert enthält.

Folgende *Mode*-IDs sind im Standard definiert[6]:

- **ProdLayout**: Mögliche Werte sind "true" und "false".
  Diese *Mode*-Anweisung gibt an, ob die Elemente bereits für die Produktion angeordnet sind. Für Fertigteilumlaufanlagen bedeutet das konkret: bei *ProdLayout=true* handelt es sich um Daten die bereits von der Palettenbelegung auf Produktions-Paletten angeordnet wurden, und bei *ProdLayout=false* handelt es sich um CAD-Daten, deren Elementkoordinaten noch nicht auf Produktionspaletten ausgerichtet sind.
  Beachte: bei *ProdLayout=true* entspricht das PXML-Dokument (die Datei) genau einer Fertigungseinheit (also z.B. einer Umlaufpalette). D.h. die Daten werden in Fertigungseinheiten "portioniert", so dass das empfangende System weiß, welche Elemente zusammen in einer Fertigungseinheit liegen.
  Anmerkung zur PTS-Prüfung: Bei *ProdLayout=false* überprüft der PTS-Server die Elemente nur einzeln und ignoriert hierbei ihre Absolutposition (der PTS-Server geht davon aus, dass die Elemente später bei der Palettenbelegung noch ideal positioniert werden). Die Drehlage der Elemente wird vom PTS-Server aber als fest vorgegeben betrachtet, d.h. er versucht *nicht* eigenständig, eine optimale Drehlage anzunehmen[7].
- **RequestedCulture:** Mit einem oder mehreren solcher Einträge kann angegeben werden, welche Landessprachen interessieren. So kann beispielsweise einem PTS-Server mitgeteilt werden, welches die Sprachen sind, in denen die Meldungstexte geliefert werden sollen. Die Einträge sind mittels ISO 639 Language Codes anzugeben, optional erweitert durch die ISO 3166 Country Codes (Beispiele: "en" oder "en-US"). Wenn mehrere Sprachen angefordert werden, sind dementsprechend mehrere *Mode*-Einträge anzuführen.
- **EstimateProdTime:** Mögliche Werte sind "true" und "false".
  Diese *Mode*-Direktive weist einen PTS-Server an, eine Produktionszeitenberechnung vorzunehmen oder nicht.
  Diese Option soll helfen, die (eventuell rechenaufwändige) Produktionszeiten-Simulation nur dann auszuführen, wenn sie tatsächlich benötigt wird.
- **EnableProduction:** Mögliche Werte sind "true" und "false".
  Diese *Mode*-Direktive deklariert, dass für die betreffenden Daten eine uneingeschränkte Produktionsfreigabe besteht. Diese Direktive dient zum Unterscheiden von endgültigen CAD-Daten (mit Produktionsfreigabe) und Vorabversionen derselben Daten.
- **EnableReinforcement:** Mögliche Werte sind "true" und "false".
  Diese *Mode*-Direktive ist ähnlich der *EnableProduction*-Direktive, schränkt aber ein, dass die Freigabe für die Bewehrungsproduktion gilt. Diese Direktive ist insbesondere bei Anlagen nützlich, die einen längeren Vorlauf bei der Bewehrungsvorbereitung erfordern.

---

[6] Darüber hinaus steht es jeder Applikation frei, für interne Zwecke eigene zusätzliche *Mode-ID*s zu definieren. Diese sollten dann mit "I_" beginnen, um Konflikte mit eventuellen späteren Erweiterungen des Standards auszuschließen.

[7] Es wird bewusst eine unterschiedliche Behandlung von Positionsoffset und Drehlage festgelegt: ersterer wird bei *ProdLayout=false* nicht geprüft, letztere dagegen immer. Es wäre praxisfern zu verlangen, dass ein CAD-System die Platten für die PTS-Anfrage produktionstechnisch optimal positioniert – denn das ist wesentliche Aufgabe des Palettenbelegungsprozesses. Andererseits würde es nicht funktionieren, den PTS-Server alle möglichen Drehlagen eigenständig durchprobieren zu lassen. Denn einerseits weiß der PTS-Server zu wenig über die Praktikabilität einer Drehung der Elemente, und andererseits kann es bei Verkettung von Systemen zu falschen PTS-Testergebnissen kommen, wenn unterschiedliche Teilsysteme unterschiedliche Drehlagen annehmen. (Dieses Argument könnte im Prinzip zwar auch für die Positionierungsoffsets angeführt werden, doch hat es dort geringe Praxisrelevanz).

Die Bewehrungsproduktion muss dann eventuell schon vor der definitiven Zeichnungsfreigabe beginnen.

- **EnableProcurement:** Mögliche Werte sind "true" und "false".
  Diese *Mode*-Direktive ist ähnlich der *EnableProduction*-Direktive, schränkt aber ein, dass die Freigabe für die Materialbeschaffung gilt.

Beispiel für *Mode*-Angaben:

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
  <DocInfo GlobalID="7C7E1FC5-0A46-48c5-A3B2-249D75B70BCF">
    <MajorVersion>1</MajorVersion>
    <MinorVersion>3</MinorVersion>
    <Mode>
      <ID>ProdLayout</ID>
      <Val>true</Val>
    </Mode>
    <Mode>
      <ID>RequestedCulture</ID>
      <Val>en</Val>
    </Mode>
    <Mode>
      <ID>RequestedCulture</ID>
      <Val>fr-BE</Val>
    </Mode>
    <Mode>
      <ID>EstimateProdTime</ID>
      <Val>false</Val>
    </Mode>
  </DocInfo>
</PXML_Document>
```

3.3 Order

Der *Order*-Abschnitt kann beliebig oft vorkommen (auch 0 mal). Er enthält Fertigungseinheiten mit übereinstimmenden Auftragsdaten.

Beachte: ein *Order*-Abschnitt muss *nicht alle* Fertigungseinheiten eines Auftrages enthalten. Er darf jedoch nicht Fertigungseinheiten zu verschiedenen Aufträgen enthalten.

Im Order-Abschnitt sind im Wesentlichen jene Informationen untergebracht, die unabhängig von der genauen Element-Aufteilung sind. Diese Daten sind demnach typischerweise unabhängig von den Details der Fertigteil-Detail-Zeichnung.

3.3.1 Order Information

OrderNo: Auftragskennung.

Structure: Gruppe von Gebäuden, z.B. Erste Zeile einer Reihenhaus-Siedlung.

Building: Einzelgebäude.

Storey: Geschossbezeichnung.

SubStorey: Teilbereich eines Geschosses.

Component: Bauteil.

DrawingNo: Kennung der Zeichnung.

DrawingDate: Datum an dem die Zeichnung erstellt oder geändert wurde.

DrawingRevision: Revisionsnummer der Zeichnung.

DrawingAuthor: Ersteller der Zeichnung.

ErpProjectUnit: Teilprojektbezeichnung im ERP. Stimmt oft mit *DrawingNo* überein. Während *DrawingNo* aber einer im CAD festgelegten Einteilung folgt, bezieht sich *ErpProjectUnit* auf eine Projektstrukturierung im ERP.

DeliveryDate: Geplantes Lieferdatum.

GenericOrderInfo01[8]: Bauvorhaben – Name

GenericOrderInfo02: Bauvorhaben – Allgemeines

GenericOrderInfo03: Bauvorhaben – Adresse

GenericOrderInfo04: Bauvorhaben – Info

GenericOrderInfo05: Bauort – Name

GenericOrderInfo06: Bauort – Straße

GenericOrderInfo07: Bauort – PLZ

GenericOrderInfo08: Bauort – Ort

GenericOrderInfo09: Bauherr – Name

GenericOrderInfo10: Bauherr – Straße

GenericOrderInfo11: Bauherr – PLZ

GenericOrderInfo12: Bauherr – Ort

GenericOrderInfo13..20: Individuell festzulegende Zusatzdefinitionen.

Comment: Beliebiger Kommentar zum Auftrag.

OrderArea: Verrechnungsfläche in m^2 zum gesamten Auftrag (also bezogen auf die betreffende *OrderNo*). Dieser Wert wird typischerweise in der Kommunikation von CAD nach ERP verwendet.

3.3.2 Import Source Information

ImportSource: Falls der Auftrag aus einer anderen Datei importiert wurde, kann in *ImportSource* der Dateiname angegeben werden.

ImportSourceType: In *ImportSourceType* kann bei importierten Daten der Typ der Ursprungsdatei angegeben werden (z.B. BVBS oder Unitechnik 5.2).

3.3.3 ApplicationName, ApplicationGUID, ApplicationVersion

Diese Angaben identifizieren die Applikation, mit der die Daten erstellt oder verändert wurden. Im Zweifelsfall bestimmt dies, wie die Daten zu interpretieren sind.

Typischerweise werden diese Angaben beim Laden aus Datei oder Zwischenablage ausgewertet, und dann (nach einer eventuellen Daten-Konvertierung) auf null gesetzt. Applikationsintern sollten diese Werte nicht verwendet werden.

Der **ApplicationName** ist typischerweise der Titel der Applikation, ohne genaue Versionsangabe und ohne Anspruch auf Eindeutigkeit.

Die **ApplicationGUID** sollte die Applikation eindeutig identifizieren, jedoch ebenfalls ohne genauere Versions-Angabe (eventuell könnte man für jede Hauptversion eine eigene GUID vergeben). Die GUID sollte das Format "59303B9F-B7E1-42bf-857A-9F6574A37433" haben.

Die **ApplicationVersion** soll eine möglichst genaue Versionsangabe beinhalten; typischerweise ist dies eine Zeichenkette der Form "4.21.2471.40371".

[8] Beim UNICAM-Import und -Export werden die ersten 12 GenericOrderInfo-Einträge (sofern vorhanden) auf die 12 Zeilen von Bauvorhaben/Bauort/Bauherr abgebildet.

3.4 OrderInfo, OrderInfoVal

[Entworfen in Zusammenarbeit mit Precast Software Engineering GmbH, Salzburg (A).]

Die **OrderInfo**-Tabelle enthält Zusatzeinträge zum *Order*-Block. Die Art der Einträge ist applikationsabhängig und wird über das **Type**-Attribut unterschieden.

In *OrderInfo*-Einträgen können beispielsweise umfangreiche Projektvorgaben beschrieben werden.

Die *OrderInfo*-Tabelle hat nur ein **Code**-Feld: ein alphanumerischer Code im Sinne eines Artikelcodes oder eines anderen Matchcodes. Die Bedeutung des **Codes** ist nur innerhalb eines *OrderInfo*-Typs definiert. D.h. unterschiedliche *OrderInfo*-Typen haben voneinander unabhängige **Code**-Systematiken.

Die **OrderInfoVal**-Tabelle ist eine Untertabelle zur OrderInfo. Jeder *OrderInfoVal*-Untereintrag hat ebenfalls ein **Type**-Attribut, das die Bedeutung des Untereintrags festlegt. Im **V**-Attribut wird ein alphanumerischer Wert angegeben. Optional können im *OrderInfoVal*-Eintrag noch ein **Culture**-Attribut (Sprache) und ein **U**-Attribut (Einheit) angegeben werden[9]

Im Standard sind folgende Typen definiert[10]:

- OrderInfo Type = "**DrawingTemplate**":
  Ein „DrawingTemplate"-Eintrag legt fest, welche Projektvorlage beim Erstellen einer Detailzeichnung verwendet werden soll. Dadurch kann beispielsweise das ERP-System bei der Zeichnungserstellungs-Beauftragung entsprechende vorgaben an das CAD machen. Der *OrderInfo*-Eintrag selbst enthält nur einen *Code*, der die Vorlage identifiziert. In Abhängigkeit der gewählten Vorlage kann es notwendig sein, weitere Angaben in *OrdeInfoVal*-Untereinträgen zu liefern.
  - OrderInfoVal Type = "**LocalizedName**": Eine lokalisierte Bezeichnung.
  - OrderInfoVal Type = "**Description**": Ein lokalisierter Beschreibungstext..
- OrderInfo Type = "**AccountingPosition**":
  Ein „AccountingPosition"-Eintrag beschreibt eine Abrechnungsposition im Auftrag. Durch die Übertragung von Abrechnungspositionsbeschreibungen vom ERP zu CAD können im CAD die zu zeichnenden Elemente konkreten Vertragsabrechnungspositionen zugeordnet werden. Der *OrderInfo*-Eintrag selbst enthält nur den *Code* der Abrechnungsposition (die ID der Abrechnungsposition. Sie ist es, die dann bei der Übertragung CAD→ERP im Feld *Product.ItemPosition* referenziert wird). Die Details der Abrechnungsposition können in *OrdeInfoVal*-Untereinträgen beliebig detailliert beschrieben werden.
  - OrderInfoVal Type = "**LocalizedName**": Eine lokalisierte Bezeichnung.
  - OrderInfoVal Type = "**Description**": Ein lokalisierter Beschreibungstext..
  - OrderInfoVal Type = "**PositionInBOQ**": Bezeichnung der entsprechenden Leistungsverzeichnispositionen.
  - OrderInfoVal Type = "**PositionInOrder**": Bezeichnung der entsprechenden Auftragsposition.
  - OrderInfoVal Type = "**PSE.AccPos.XYZ**": Eine Abrechnungspositionsinformation, wie sie von Precast Software Engineering Systemen unter der Bezeichnung XYZ geführt wird.

Beispiel:

```
<OrderInfo Type="AccountingPosition">
  <Code>U1.F1.Stairs</Code>
  <OrderInfoVal Type="LocalizedName" V="Geschoss 1, Treppen" Culture="de"/>
  <OrderInfoVal Type="LocalizedName" V="Floor 1, Stairs" Culture="en"/>
  <OrderInfoVal Type="PSE.AccPos.IncludedReinforcement" V="0.23" U="kg/m²"/>
</OrderInfo>
```

[9] Bezüglich Einheitenangabe gilt dasselbe wie für die Unit des ElemenInfo-Eintrages.

[10] Zusätzlich kann es weitere applikationsspezifische Typen geben. Die entsprechenden applikationsspezifischen *Type*-Werte sollen mit vorangestelltem "I_" beginnen, um Konflikte mit späteren Erweiterungen des Standards auszuschließen.

3.5 Product (Element)

Bei Doppelwänden entspricht das **Product** der vollständigen Wand; es enthält beide Wandschalen.

Die Informationen der UNICAM-SLABDATE sind teilweise im Product-Abschnitt und teilweise im Slab-Abschnitt untergebracht.

3.5.1 ElementNo

[Ersetzt *Slab.SlabNo*, siehe Abschnitt 3.7.7]

Element-Bezeichnung.

3.5.2 ProductType

Entspricht in etwa der Produkt-Spezifikation des UNICAM-HEADER.

Man kann beliebige Typangaben verwenden. Folgende Typen sind fest definiert:

- **00:** Lattice girder floor and roof slab
- **DW:** Double wall
- **03:** Prestressed slab
- **04:** Isolating slab
- **05:** Facade element
- **06:** Solid floor
- **07:** Silo slab
- **08:** Constructional part
- **09:** Solid wall
- **10:** Hollo core slab
- **11:** Sandwich panel
- **16:** Brick floor
- **19:** Brick wall
- **NW:** Zero wall
- **TW:** Thermo wall
- **36:** Light-weight concrete full-thickness floor
- **39:** Light-weight concrete solid wall
- **GML:** Generic Multi Layer element
- **BM:** Beam
- **CL:** Column
- **ST:** Stairs
- **MD:** Module
- **DT:** Double-tee
- **InSitu:** In-situ concrete element. Elemente dieses Typs sind nicht vorgefertigte Fertigteile, sondern in Ortbeton gegossene Teile des Gebäudes. Typisches Beispiel: obere Lage von Decken.

Beachte: falls in einem Order-Block verschiedene ProductType-Werte vorkommen, wird der Order-Block beim UNICAM-Export auf mehrere HEADER-Blöcke aufgeteilt (In UNICAM steht der Product-Type im HEADER-Block).

Beachte: für die Doppelwand werden hier *nicht* die Typen 01 und 02 verwendet, sondern der Typ **DW**; Im PartType des Slabs wird dann zwischen Wandhälfte 1 und Wandhälfte 2 unterschieden.

Virtual Element

Wenn kein *ProductType* angegeben wird (oder ein Leerstring angegeben wird), handelt es sich um ein **Virtuelles Element**: ein solches Element dient zum Übertragen von Projektbezogenen Zusatzinformationen, die keinem realen Element zugeordnet werden können. Beispiele sind projektbezogene Zusatzleistungen, die aus Verrechnungsgründen angeführt werden, oder Zusatzmaterialien, die keinem einzelnen Element zugeordnet werden können.

3.5.3 PieceCount

Soll-Stückzahl.

3.5.4 Datenübergabe für Doppelwände: TurnWidth, TotalThickness, DoubleWallsGap

Für Doppelwände (und andere doppelwandartige-Elemente) gilt folgende Festlegung:

Die Einzelplatten werden in ihrer Drehlage so dargestellt, wie sie vor dem Einwenden auf den Einzelpaletten liegen; die erste (= die einzuwendende) Wandhälfte ist also gegenüber dem fertigen Produkt gedreht darzustellen.

Die Darstellung der Einzelplatten entspricht also der produktionstechnischen Situation, und nicht jener des fertigen Produktes. Die Doppelwand wird also in *aufgeklappter* Form beschrieben.

Um zu verstehen, wie das fertige Element aussieht, reicht es nicht aus, die Einzelplatten zu betrachten, sondern man muss zusätzlich wissen, wie die beiden Wandhälften zusammengeführt werden. Das erfolgt über die Angabe von **TotalThickness** und **TurnWidth**[11].

DoubleWallsGap beschreibt den inneren Spaltabstand zwischen den beiden Wandhälften in eingewendeter Form. Tatsächlich ist dieser Wert redundant, da er sich aus anderen Werten rechnerisch ergibt. Diesem Wert kommt daher keinerlei fundamentale Bedeutung zu und er wird nur aus Kompatibilitätsgründen mitgeführt.

Eine mathematisch-formalere Beschreibung der geometrischen Darstellung von Doppelwänden erfolgt im Abschnitt 3.7.4.

Hinweis zum Import von UNICAM-Daten aus CAD-Systemen:

Wenn UNICAM-Daten von einem CAD-System übernommen werden, muss man eine Annahme über die verwendeten Doppelwand-Koordinaten treffen. Je nach CAD-System, und je nach CAD-Systemeinstellung, kann es hier ganz unterschiedliche Festlegungen geben, die aber aus dem UNICAM-File selbst nicht ersichtlich sind. Die häufigsten Konventionen sind:

a) Die Doppelwandhälften werden in **aufgeklappter Darstellung** übertragen, also sowie in PXML. Da es in UNICAM kein *TurnWidth*-Feld gibt, muss man hierbei wissen, welche Palettenbreite im CAD angenommen wird; andernfalls sind die Daten nicht korrekt interpretierbar. Sinnvoll, aber leider ungebräuchlich, wäre es, in UNICAM die aufgeklappte Form unter der Annahme *TurnWidt*=0 darzustellen.

b) Die zweite Wandhälfte wird so wie im Fall a) übergeben, und damit gleich wie im PXML. Die erste Wandhälfte wird dagegen in **pseudo-gewendeter Form** übergeben: die Daten entstehen aus der aufgeklappten Form durch Drehen um 180° um die Achse *Y*=*TurnWidth*/2 (=Einwenden) und anschließendes Spiegeln um die XY-Ebene.

c) Die **invers pseudo-gewendete Form** ist ähnlich dem Fall b), wobei aber die Doppelwand von unten betrachtet wird, d.h. mit Blick auf die Außenseite der zweiten Wandhälfte. Beim Importieren dieser Daten muss man die gesamte Wand um 180° um die Z-Achse drehen, und dann noch die zweite Wandhälfte pseudo-wenden.

Die *pseudo-gewendete Form* und *invers pseudo-gewendete Form* sind durch eine sehr einfache Transformation ***Myz*** miteinander verknüpft:

$$M_{yz} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

[11] Die Wahl von *TurnWidth* ist eigentlich frei und muss lediglich beim setzten von *Slab.Y* und *Slab.ProdY* korrekt berücksichtigt werden. Optimal ist es, *TurnWidth* auf 0 zu setzten.

Diese Transformation beschreibt eine Spiegelung an der *YZ*-Ebene und sie kann verwendet werden, um die *pseudo gewendete Form* in die *invers pseudo-gewendete Form* zu transformieren und umgekehrt (*Myz* ist eine selbsinverse Matrix, d.h. Transformation und Umkehrtransformation sind identisch). Zudem kann man *Myz* auf eine "falsch" transformierte Form anwenden, um das korrekte Ergebnis zu erhalten: wenn man auf eine invers *pseudo-gewendete Form* jene Transformation anwendet, die eigentlich für die *pseudo-gewendete* Form korrekt wäre, erhält man eine „falsche" *aufgeklappte Darstellung*. Diese kann man mittels *Myz* in die korrekte *aufgeklappte Darstellung* bringen (dasselbe gilt auch umgekehrt, also wenn man von der *pseudo-gewendeten Form* ausgeht).

In diesem Sinne ist *Myz* eine sehr einfache und universelle Transformation um zwischen *pseudo-gewendeter* und *invers pseudo-gewendeter* Form hin und her zu schalten.

3.5.5 Comment

Beliebiger Kommentar zum Produkt.

3.5.6 RotationPosition

Ein Winkel in Grad, der angibt, wie das Element gegenüber der ursprünglichen CAD-Zeichnung gedreht wurde (Die Drehung erfolgt typischerweise während des Palettenbelegungsprozesses aus produktionstechnischen Gründen).

Die Drehung des Elementes erfolgt um die Z-Achse im positiven Umlaufsinn (also gegen Uhrzeigersinn). Da die erste Wandhälfte von Doppelwänden vor dem Einwenden in geklappter Form vorliegt, muss der Winkel dort in negativem Umlaufsinn gerechnet werden.

Es wird empfohlen, die Elemente nicht gedreht zu übergeben und folglich auch die *RotationPosition* nicht zu verwenden. An Stelle der gedrehten Übergabe der Elemente sollte man von *Slab.ProdRotX/Y/Z* Gebrauch machen[12].

3.5.7 Stacking Information

StackNo:

[Ersetzt *Slab.NumberOfMeansOfTransport*, siehe Abschnitt 3.7.7]

Identifikationskennung des Transportstapels.

Die Stapelnummer ist vom Datentyp her ein Text und kann auch strukturiert sein. So können beispielsweise zwei Stapel, die gemeinsam transportiert werden sollen, in über eine Bezeichnungsstruktur der Form „17.1" und „17.2" zusammengehalten werden.

StackID:

Eindeutige Datenbank-ID des Stapels. Diese ID identifiziert den Stapel auftragsübergreifend in eindeutiger Weise.

StackingSequence:

[Ersetzt *Slab.TransportSequence*, siehe Abschnitt 3.7.7]

Reihenfolge innerhalb eines Transportstapel.

Das Element mit dem kleinsten Wert muss als erstes dem Stapel hinzugefügt werden[13].

---

[12] Eine Schwierigkeit bei der Verwendung der *RotationPosition* ist, dass man die Lage des ursprünglichen (nicht gedrehten) Elementes „erraten" muss, um die Projektkoordinaten und die Stapelkoordinaten richtig zuordnen zu können, denn Projekt- und Stapelkoordinaten beziehen sich auf das ursprüngliche (nicht gedrehte) Element. Die im CAD durchgeführte Dreh-Schiebung ist aber nicht eindeutig bekannt, da zwar die Drehung in *RotationPosition* angegeben ist, aber nicht die Verschiebung. Die genaue Vorschrift für die Rückrechnung hängt somit vom CAD ab. In vielen Fällen wird es so sein, dass man das Element um den in *RotationPosition* angegebenen Wert zurückdrehen muss und dann so verschieben muss, dass das umhüllende Rechteck der Betonkonturen bei (0, 0) liegt.

[13] Üblicherweise entspricht das *StackingSequence*-Feld dem UNICAM-Feld "Montagereihenfolge im Transportstapel". Laut UNICAM-Definition ist letzteres aber gegenläufig zu *StackingSequence* definiert. Die meisten CAD-Systeme

StackingLevel:

[Ersetzt *Slab.PileLevel*, siehe Abschnitt 3.7.7]

Stapelebene innerhalb des Transportstapels. Dieses Feld ist nur dann interessant, wenn es auf einer Stapelebene mehrere nebeneinander liegende Elemente geben kann. Andernfalls ist die *StackingSequence*-Angabe ausreichend.

Zudem hat dieses Feld nur Informationscharakter für den Maschinenbediener. Eine automatische Abstapelvorrichtung wird die exakten Stapelkoordinaten (siehe unten) verwenden.

StackingX/Y/Z/Angle/RotY/RotX:

[Ersetzt *Slab.PositionInPileX/Y/Z* und *Slab.AngleInPile*, siehe Abschnitt 3.7.7]

Lage des Elementes auf dem Transportstapel.

Die Lage des Elementes auf dem Stapel wird ermittelt indem folgende Transformationen in der hier gegebenen Reihenfolge ausgeführt werden:

1) Drehung um *StackingAngle* um die absolute Z-Achse.
2) Drehung um *StackingRotY* um die absolute Y-Achse.
3) Drehung um *StackingRotX* um die absolute X-Achse.
4) Verschiebung um *StackingX/Y/Z*

(Alle Winkel in Grad).

3.5.8 Project Coordinates

[Ersetzt *Slab.ProjectCoordinates*, siehe Abschnitt 3.7.7]

Die Projektkoordinaten beschreiben die Lage des Elementes im reale Gebäude.

Es werden 3 Punkte P_1, P_2 und P_3 angegeben, die folgende Bedeutung haben:

- P_1 bezeichnet den Raumpunkt im Gebäude, an dem das Elementes zu liegen kommt.
- Der Vektor $\overrightarrow{P_1P_3}$ bezeichnet jene Achse im Gebäude, die der X-Achse des Elementes entspricht.
- Der Vektor $\overrightarrow{P_1P_2}$ bezeichnet jene Achse im Gebäude, die der Y-Achse des Elementes entspricht[14].

Anmerkung:

Projektkoordinaten beschreiben eine Drehung und Verschiebung im Raum. Die etwas eigentümliche Darstellung mittels der genannten 3 Punkte wurde aus Kompatibilität zu UNICAM gewählt. Tatsächlich ist es aber einfacher und üblicher, die Transformation durch Drehmatrix R und Verschiebevektor $\vec{t}$ zu beschreiben. Ein Punkt der Elementdarstellung wird dann in die Gebäudekoordinaten dadurch übergeführt, dass er zunächst mittels R gedreht und anschließend mittels $\vec{t}$ verschoben wird.

Bei gegebenem R und $\vec{t}$ kann man über folgende Beziehungen P_1, P_2, P_3 ermitteln (Man beachte aber, dass P_1, P_2, P_3 nicht eindeutig sind):

$$P_1 = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}, \qquad P_2 = \begin{pmatrix} R_{12} + t_x \\ R_{22} + t_y \\ R_{32} + t_z \end{pmatrix}, \qquad P_3 = \begin{pmatrix} R_{11} + t_x \\ R_{21} + t_y \\ R_{31} + t_z \end{pmatrix}$$

Umgekehrt können bei gegebenen P_1, P_2, P_3 die Transformationen R und $\vec{t}$ ermittelt werden:

---

beschreiben das UNICAM-Feld aber – entgegen der offiziellen Definition – gleich wie die hier definierte *StackingSequence*.

[14] Diese Festlegung setzt voraus, dass $\overrightarrow{P_1P_2}$ orthogonal zu $\overrightarrow{P_1P_3}$ ist. Das wird auch meistens erfüllt sein, kann aber natürlich nicht garantiert werden. Die mathematisch allgemeinere Festlegung ist daher so, dass die Y-Achse durch das doppelte Kreuzprodukt $(\overrightarrow{P_1P_3} \times \overrightarrow{P_1P_2}) \times \overrightarrow{P_1P_3}$ gegeben ist.

$$\vec{t} = \begin{pmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \end{pmatrix}$$

$$\begin{pmatrix} R_{11} \\ R_{21} \\ R_{31} \end{pmatrix} = \left(\overrightarrow{P_1P_3}\right)^0, \qquad \begin{pmatrix} R_{12} \\ R_{22} \\ R_{32} \end{pmatrix} = \left(\left(\overrightarrow{P_1P_3} \times \overrightarrow{P_1P_2}\right) \times \overrightarrow{P_1P_3}\right)^0, \qquad \begin{pmatrix} R_{13} \\ R_{23} \\ R_{33} \end{pmatrix} = \begin{pmatrix} R_{11} \\ R_{21} \\ R_{31} \end{pmatrix} \times \begin{pmatrix} R_{12} \\ R_{22} \\ R_{32} \end{pmatrix}$$

Hier wurde folgende Notation den Einheitsvektor verwendet:

$$(\vec{a})^0 := \frac{\vec{a}}{|\vec{a}|}$$

3.5.9 Supplementary Product Information

AdditionInfo

[Ersetzt *Slab.ProductAddition*, siehe Abschnitt 3.7.7]

Zusatzinformation zum Produkt.

UnloadingInfo

[Ersetzt *Slab.TypeOfUnloading*, siehe Abschnitt 3.7.7]

Entladeart.

TransportInfo

[Ersetzt *Slab.MeansOfTransport*, siehe Abschnitt 3.7.7]

Informationen zum Transport nach erfolgter Produktion (Transportmittel, Transport-Gruppierung).

Hier kann der Typ des Transport-Containers angegeben werden.

ItemPosition

[Ersetzt *Slab.OrderPosition*, *Slab.ProductGroup*, *Slab.SlabType* und *Slab.ItemDesignation*, siehe Abschnitt 3.7.7.
Genauer gesagt, gilt:
Product.ItemPosition = *Slab.OrderPosition@Slab.ProductGroup@Slab.SlabType@Slab.ItemDesignation*]

Produktionspositionskennung die typischerweise auf Referenzen im ERP-System Bezug nimmt.

Der Wert des Felder kann (aber muss nicht) eine mehrteilige Struktur aufweisen. Die einzelnen Teile sind dann durch @-Zeichen voneinander zu trennen (Beispiel: „123@AFX5“).

Eine konkrete Verwendung dieses Feldes besteht in der Angabe der ERP-Abrechnungs-Position, der das Element zugeordnet ist. Hier wird typischerweise auf ID-Werte Bezug genommen, die zuvor vom ERP über *Order.OrderInfo*-Einträge (vom Typ *AccountingPosition*) an das CAD übermittelt wurden.

3.6 ElementInfo

[Entworfen in Zusammenarbeit mit Precast Software Engineering GmbH, Salzburg (A).]

Die *ElementInfo*-Tabelle enthält Zusatzeinträge zum *Product*-Block. Die Art der Einträge ist applikationsabhängig und wird über das **Type**-Attribut unterschieden.

Die *ElementInfo*-Einträge eignen sich besonders, um elementspezifische Informationen vom CAD-System zum ERP-System zu übertragen. Hier werden typischerweise zusammenfassende Mengen- und Artikelangaben benötigt, ohne bis zum letzten geometrische Detail hinunterzugehen. Dementsprechend geht die CAD-zu-ERP-Datenübertragung oft nur bis zur Tiefe der *Product*-Ebene; Details der darunter liegenden Strukturen werden in *ElementInfo*-Einträgen zusammenfassend betrachtet.

Auch an das Produktionssystem (Produktionsleitsystem) können über *ElementInfo*-Einträge Zusatzinformationen übertragen werden, die ergänzend zu den eigentlichen (geometrisch exakten) Produktionsdaten sind. So kann es beispielsweise für das Produktionssystem von Bedeutung sein,

neben den geometrischen Produktdetails auch Abrechnungsmaße zu kennen, um Produktionsstatistiken und Solltakt-Vorgaben korrekt anzeigen zu können.

Das Attribut **Inventory** gibt an, ob der *ElementInfo*-Eintrag für die Warenwirtschaft vorgesehen ist. Die eingetragenen Mengenangaben sind dann warenwirtschaftlich genau und können für Lagerverwaltungsaufgaben verwendet werden. Wenn das *Inventory*-Feld nicht gesetzt wird, oder auf *false* steht, handelt es sich häufig um konsolidierte Mengenangaben, die eher für eine buchhalterische Abrechnung vorgesehen sind.

3.6.1 Felder der ElementInfo-Einträge

Die genaue Bedeutung der einzelnen Felder ist abhängig vom *Type*-Attribut. Eine grobe Bedeutung der Felder kann aber typunabhängig beschrieben werden:

- **Code:** ein alphanumerischer Code im Sinne eines Artikelcodes oder eines anderen Matchcodes. Der PXML-Standard macht für die *Code*-Werte keine Vorgaben. D.h. die Codes sind anlagenspezifisch zwischen den CAD- und ERP-Systemen zu vereinbaren[15].
- **Description:** Eine textartige Beschreibung oder Benennung.
- **ObjectID:** Identifikation eines Objektes, auf das sich der *ElementInfo*-Eintrag bezieht. Es können sich auch mehrere *ElementInfo*-Einträge auf ein- und dasselbe Objekt beziehen. Ein solcherart referenziertes Objekt kann durchaus auch elementübergreifend sein (z.B. eine elementübergreifende Aussparung).
- **PieceCount:** Ein ganzzahliger Multiplikator für den gesamten *ElementInfo*-Eintrag. Wenn der Wert nicht angegeben ist, wird normalerweise eine Stückzahl von 1 angenommen. Wenn *PieceCount* einen Wert von *n* hat, ist das genauso zu werten, als ob der *ElementInfo*-Eintrag *n* mal (mit jeweils Stückzahl 1) angeführt worden wäre.
- **Val1, Val2:** Mengen- oder Größenangaben. Die genaue Bedeutung sowie die zugrundeliegende Maßeinheit werden *Type*-abhängig vorgegeben.
  Wenn der *ElementInfo*-Eintrag einen *PieceCount*-Wert größer 1 hat, beziehen sich die Mengenangaben auf nur 1 Stück.
- **Unit:** Maßeinheit für die Werte *Val1* und *Val2*. Folgende Werte sind im Standard vorgesehen: ***n*** (= Stückzahl), ***mm***, ***cm***, ***m***, ***km***, ***m²***, ***m³***, ***L*** (=Liter), ***kg***, ***t***, ***EUR*** (sowie alle anderen Währungskürzel nach ISO 4217). Wenn keine Maßeinheit angegeben wird, sind SI-Basiseinheiten zu verwenden (m, m², m³, kg). Zusammengesetzte Maßeinheiten werden mit „*“ und „/“ verbunden: z.B. **kg/m²**, **N*m**.
- **Details:** Abhängig vom jeweiligen *Type*-Wert können hier weitere Details angegeben werden. In komplexen Fällen kann hier auch nochmal eine ganze Datenstruktur in Form eines XML-Strings angegeben werden.

Zusätzlich kann es weitere applikationsspezifische Felder geben (I_P_-Felder).

3.6.2 Vordefinierte ElementInfo-Typen

Im Folgenden werden die *ElementInfo*-Typen beschrieben, die derzeit im Standard definiert sind[16]:

- ➢ ElementInfo Type = "**AccArea**":
  Abrechnungsfläche des Elementes.
  Die Fläche des Elementes wird hier aufgrund vereinbarter Abrechnungskonventionen angegeben. Dieser Wert kann durchaus von der realen geometrischen Fläche abweichen.
  Val1: Wert der Fläche in m^2.

[15] Durch diese "Freiheit" grenzt sich die Rolle des *Codes* deutlich von der Rolle des *Type*-Wertes ab.

[16] Zusätzlich kann es weitere applikationsspezifische Typen geben. Die entsprechenden applikationsspezifischen *Type*-Werte sollen mit vorangestelltem "I_P_" beginnen, um Konflikte mit späteren Erweiterungen des Standards auszuschließen.

- ElementInfo Type = "**AccAreaExt**":
  Erweiterte Abrechnungsfläche des Elementes.
  Der Typ ähnelt jenem des "AccArea", betrachtet aber eine durch Überstände (meist Eisenüberstände) erweiterte Fläche.
  Val1: Wert der Fläche in m^2.
- ElementInfo Type = "**AccAreaAdd**", "**AccAreaExtAdd**":
  Zusatzabrechnungsfläche des Elementes.
  Dier kann eine weitere Zusatzabrechnungsfläche angegeben werden, die noch nicht in *AccArea* oder *AccAreaExt* enthalten ist.
  Val1: Wert der Fläche in m^2.
- ElementInfo Type = "**EffortArea**":
  Aufwandsfläche des Elementes.
  Eine in m^2 angegebene Aufwandsabschätzung für das Element. Die Berechnung basiert meist auf einer realen oder buchhalterischen Fläche, vermehrt um Anteile, die Schwierigkeitsgrade und Aufwände wiederspiegeln. Der so entstehende fiktive Flächenwert soll weitgehend den realen Wert des Elementes wiederspiegeln. Die produzierte Aufwandsfläche ist ein guter Indikator für die effektive Produktionsleistung einer Anlage.
  Val1: Wert in m^2.
- ElementInfo Type = "**ArchitecturalPart**":
  Architektonische Einheit (z.B. Wand oder Decke), die dem Element übergeordnet ist. Elemente, die zur selben übergeordneten architektonischen Einheit gehören, sollen eine gemeinsame *ObjectID* haben.
  ObjectID: Identifikation der übergeordneten architektonischen Einheit.
  Code: Artikelcode der den Typ der architektonischen Einheit beschreibt.
  Val1: Gesamtfläche der architektonischen Einheit in m^2. (abrechnungstechnischer Wert, typischerweise inklusive Aussparungen).
  Val2: Teilfläche (in m^2) der architektonischen Einheit, die vom betreffenden Element abgedeckt wird (abrechnungstechnischer Wert, typischerweise inklusive Aussparungen). Dieser Wert ist eventuell identisch mit *AccArea*; dann kann er auch entfallen.
- ElementInfo Type = "**Outlet**":
  Aussparung im Decken- oder Wandprojekt.
  Eine Aussparung kann auch elementübergreifend sein und kann daher in *ElementInfo*-Einträgen unterschiedlicher Elemente referenziert werden. Es ist daher wichtig, die zugrundeliegende Aussparung mittels *ObjectID* zu identifizieren[17].
  Val1: Gesamtfläche der Projektaussparung in m^2.
  (Alternativ kann in *Val1* auch eine Stückzahl von aussparungen angegeben werden. Dann muss aber *Unit* auf „*n*“ gesetzt werden.)
  Val2: Teilfläche der Aussparung (in m^2), die im betreffenden Element liegt.
  ObjectID: Eindeutige Identifikation der Projektaussparung im CAD. Wenn die Flächensumme aller Aussparungen eines Projektes gebildet werden soll, dürfen die Aussparungen zu einer *ObjectID* nur einmal gezählt werden.
  Falls die Aussparung auch einer architektonischen Einheit (*ArchitecturalPart*) zugeordnet werden soll, sollte die *ObjectID* folgende Struktur haben:
  abc.xyz
  Hierbei ist *abc* die *ObjectID* der *ArchitecturalPart* und *xyz* eine Identifikation der Aussparung innerhalb der *ArchitecturalPart*.

[17] Typischerweise werden Aussparungen, die eine gewisse Größe überschreiten, abrechnungstechnisch von der Betonfläche abgezogen. Ob nun die Erkennung von "großen" und "kleinen" Aussparungen bereits im CAD-System erfolgt, oder erst im ERP-System, hängt von der konkreten Realisierung ab. Im ersten Fall würde das CAD eventuell nur die "großen" Aussparungen als "*Outlet*"-Einträge übergeben.

Code: Hier kann eine Unterscheidung zwischen verschiedenen Aussparungstypen erfolgen[18].

- ElementInfo Type = "**MountPart**":
  Einbauteil.
  Analog zum "*Outlet*" kann auch ein "*MountPart*" elementübergreifend sein. Auch hier ist die *ObjectID* daher eine unverzichtbare Kennung, wenn es darum geht, zu verhindern, ein- und dasselbe Einbauteil irrtümlich mehrfach zu zählen.
  ObjectID: Eindeutige Identifikation des Einbauteils im CAD.
  Falls das Einbauteil auch einer architektonischen Einheit (*ArchitecturalPart*) zugeordnet werden soll, sollte die *ObjectID* folgende Struktur haben:
  abc.xyz
  Hierbei ist *abc* die *ObjectID* der *ArchitecturalPart* und *xyz* eine Identifikation des Einbauteils innerhalb der *ArchitecturalPart*.
  Code: Artikelcode des Einbauteils.
  Val1, Val2: Wert-Parameter des Einbauteils. Die genaue Bedeutung ist *Code*-Abhängig.
- ElementInfo Type = "**SteelBar**":
  Rundstahl-Bewehrung.
  Code: Ein Artikel- oder Matchcode, der typischerweise einen Draht-Typ identifiziert (inklusive Drahtdurchmesser, Stahlqualität, Hersteller).
  Val1: Menge in *kg*. Abhängig vom Inventory-Feld handelt es sich hierbei um eine reale lagerverwaltungsrelevante Mengenangabe, oder um eine abrechnungsrelevante Mengenangabe. Wenn beide Aspekte interessieren, wird man dementsprechend 2 unterschiedliche Einträge vorsehen.
  Details: Hier können Details zur Verarbeitung codiert werden (z.B. Verarbeitung über flexible Mattenschweißanlage oder über Rundstahlbewehrungsroboter)[19].
- ElementInfo Type = "**LatticeGirder**":
  Gitterträger-Bewehrung.
  Code: Ein Artikel- oder Matchcode, der typischerweise einen Gitterträger-Typ identifiziert.
  Val1: Menge in *kg* (real oder abrechnungstechnisch, siehe Inventory-Feld).
- ElementInfo Type = "**StdMesh**":
  Lagermatten-Bewehrung.
  Code: Ein Artikel- oder Matchcode, der typischerweise einen Lagermatten-Typ identifiziert.
  Val1: Fläche in m^2. (real oder abrechnungstechnisch, siehe Inventory-Feld).
- ElementInfo Type = "**BentMesh**":
  Gebogene Lagermatte.
  Code: Ein Artikel- oder Matchcode, der eine gebogene Lagermatte vom Typ her so weit identifiziert, dass eine Preisfindung über Angaben in den *Val*-Feldern möglich ist.
  Val1: Gewicht in *kg* (real oder abrechnungstechnisch, siehe Inventory-Feld).
- ElementInfo Type = "**Cage**":
  Bügelkorb-Bewehrung.
  Code: Ein Artikel- oder Matchcode, der einen Bügelkorb-Typ identifiziert.
  Val1: Gewicht in *kg* (real oder abrechnungstechnisch, siehe Inventory-Feld).
- ElementInfo Type = "**Concrete**":
  Beton-Parzelle.
  Code: Ein Artikel- oder Matchcode, der die Betonmischung identifiziert.
  Val1: Volumen in m^3 (real oder abrechnungstechnisch, siehe *Inventory*-Feld).

---

[18] Wenn beispielsweise bereits im CAD die Unterscheidung von "großen" und "kleinen" Aussparungen gemacht wird, aber dennoch alle Aussparungen (große und kleine) als "*Outlet*"-Einträge angegeben werden sollen, kann die Unterscheidung zwischen groß und klein über das *Code*-Feld mitgegeben werden.
[19] Alternativ dazu könnte die Verarbeitungsinformation auch im *Code* untergebracht werden.

- ElementInfo Type = "**ErpProjectUnit**":
  ERP-Projekt-Einheit, der das Element zugeordnet ist. Hier wird typischerweise auf Werte Bezug genommen, die zuvor vom ERP über *Order.ErpProjectUnit* an das CAD übermittelt wurden.
  Code: Der Code der *ErpProjectUnit*.
- ElementInfo Type = "**PriceQty**":
  Größenangabe, die für die Wert-Berechnung des Elementes relevant ist (das kann eine Fläche, ein Volumen, eine Stückzahl oder sonst irgendeine Größe sein, die als Multiplikator für die Preisbildung verwendet wird. Im Prinzip könnte auch direkt ein Preis angegeben werden).
  Val1: Wert der preisbildenden Größe.
- ElementInfo Type = "**PlanningQty**":
  Größenangabe, die für die Produktionsplanung relevant ist. Diese Größe wird oft auch für die Beurteilung der Produktionsleistung herangezogen. Es kann durchaus vorkommen, dass die Planungsgröße eine andere Einheit hat als die Bewertungsgröße („*PriceQty*“). So könnte beispielsweise die *PriceQty* im m^3 angegeben werden und die *PlanningQty* in m oder Stück.
  Val1: Wert der produktionsplanungsrelevanten Größe.
- ElementInfo Type = "**TransportInfo**":
  Diverse Angaben zum Transportmittel. Typischerweise wird im *Code*-Feld eine Typ-Identifizierung festgelegt (Container-Typ) und in *ElemInfoVal*-Einträgen können Namen und Abmessungen angegeben werden.

3.6.3 ElemInfoVal

Falls die Felder die im *ElementInfo*-Eintrag definiert sind nicht ausreichen, können weitere Werte über *ElemInfoVal*-Einträge eingefügt werden.

Die folgende Tabelle beschreibt die *ElemInfoVal*-Typen die im Standard definiert sind.

Type	Beschreibung	Beispiel
Len	Länge	1234.5
Width	Breite	456.8
Height	Höhe	123.4
Qlty	Qualität (z.B. Materialqualität)	Q123
Name	Bezeichnung, Benennung	abcd

Es können auch noch weitere interne Typen definiert werden. Diese sollten aber den Präfix „I_“ haben.

Optional kann über ein Attribut **U** auch eine Einheit angegeben werde. Bezüglich Einheitenangabe gilt dasselbe wie für die *Unit* des *ElemenInfo*-Eintrages festgelegt wurde.

Beispiel MountPart:

```
<ElementInfo Type="Concrete">
  <Code>FK77</Code>
  <ElemInfoVal Type="Height" V="100" U="mm"/>
  <ElemInfoVal Type="Width" V="225.3" U="mm"/>
  <ElemInfoVal Type="Len" V="310.1" U="mm"/>
  <ElemInfoVal Type="Qlty" V="Q123"/>
</ElementInfo>
```


3.7 Slab (Elementteil, Platte)

3.7.1 PartType

Man kann beliebige Typangaben verwenden. Die Bedeutung des PartType hängt vom übergeordneten ProductType ab.

Für den Produkt-Typ DW sind folgende PartType-Werte fest definiert:

- 01: double wall 1st stage
- 02: double wall 2nd stage

Neben den Hauptteilen eines Elementes gibt es auch ergänzende Teile, wie beispielsweise Abschlussplatten (sogenannte „Abschaler“). Für diese ist die Kennunf 05 vorgesehen:

- 05: Supplementary part

3.7.2 Geometric Slab Placement (X/Y/Z, RotX/Y/Z)

Über Verschiebung und Drehung können die einzelnen Slabs innerhalb des Elementes geometrisch platziert werden.

Die Operationen werden in folgender Reihenfolge durchgeführt[20]:

1) Verschiebung (*X*, *Y*, *Z*)
2) Drehung *RotZ* um absolute Z-Achse
3) Drehung *RotY* um absolute Y-Achse
4) Drehung *RotX* um absolute X-Achse

Die Verschiebe-Offsets sind in mm und die Drehwinkel sind in DEG.

3.7.3 Slab Production Directives (ProdX/Y/Z, ProdRotX/Y/Z)

Bei der Datenübergabe aus dem CAD werden die Angaben *Slab.X/Y/Z*/*RotX/RotY/RotZ* nur dazu verwendet, die Position der Elementteile zueinander festzulegen. Wenn zudem festgelegt werden soll, wie die Elementteile auf der Produktionspalette anzuordnen sind, benötigt man zusätzliche **Produktions-Direktiven**, die in den *Slab*-Feldern **ProdX**, **ProdY**, **ProdZ**, **ProdRotX**, **ProdRotY**, **ProdRotZ** anzugeben sind. Für die Positionierung auf der Produktionspalette sind folgende Transformationen in der hier angeführten Reihenfolge durchzuführen:

1) Drehung *ProdRotX* um absolute X-Achse
2) Drehung *ProdRotY* um absolute Y-Achse
3) Drehung *ProdRotZ* um absolute Z-Achse
4) Verschiebung (*ProdX*, *ProdY*, *ProdZ*)

Insgesamt ergibt sich die Position eines Elementteils auf der Produktionspalette durch Hintereinandrausführung der folgenden Transformationen:

1) *Slab.X/Y/Z*
2) *Slab.RotZ*
3) *Slab.RotY*
4) *Slab.RotX*
5) *Slab.ProdRotX*
6) *Slab.ProdRotY*
7) *Slab.ProdRotZ*

[20] Die Reihenfolge der Operationen ist exakt einzuhalten. Die gewählte Festlegung mag etwas ungewöhnlich wirken und ist durch Kompatibilitätsüberlegungen motiviert.

8) *Slab.ProdX/ProdY/ProdZ*

3.7.4 Geometric Placement und Production Directives bei Doppelwänden

Einen Sonderfall stellen die **doppelwandartigen Elemente** dar, also die Produkttypen **DW**, **NW**, **TW**. Bei diesen Elementen wird – historisch bedingt – das Slab-Placement der Wandhälfte 1 (= einzuwendende Wandhälfte) wesentlich durch *Product.TurnWidth*/*TotalThickness* bestimmt.

Im Einzelnen werden die effektiv anzuwendenden Werte für *Geometric Placement* und *Production Directives* bei der **Wandhälfte 1** wie folgt berechnet:

- $Slab1.X_{Resulting} = Slab1.X$
- $Slab1.Y_{Resulting} = Slab1.Y - Product.TurnWidth$
- $Slab1.Z_{Resulting} = Slab1.Z - Product.TotalThickness$
- $Slab1.RotZ_{Resulting} = Slab1.RotZ$
- $Slab1.RotY_{Resulting} = Slab1.RotY$
- $Slab1.RotX_{Resulting} = Slab1.RotX + 180°$
- $Slab1.ProdRotZ_{Resulting} = Slab1.ProdRotZ$
- $Slab1.ProdRotY_{Resulting} = Slab1.ProdRotY$
- $Slab1.ProdRotX_{Resulting} = Slab1.ProdRotX + 180°$
- $Slab1.ProdX_{Resulting} = Slab1.ProdX$
- $Slab1.ProdY_{Resulting} = Slab1.ProdY + Product.TurnWidth$
- $Slab1.ProdZ_{Resulting} = Slab1.ProdZ + Product.TotalThickness$

Diese Berechnungsvorschrift ist nichts anderes als die Formalisierung der Doppelwand-Regeln, wie er schon im Abschnitt 3.5.4 beschrieben wurde.

Anmerkung: Es muss betont werden, dass diese Sonderbehandlung nur für die Wandhälfte 1 erfolgt, also für ein Elementteil mit *PartType* „1“ oder „01“ (und auch das nur bei den Produkttypen DW, NW, TW). Falls man diese Sonderbehandlung vermeiden will, und stattdessen ein DW-Element mit direkter Placement-Angabe definieren will, kann man den *PartType* einfach leer lassen – die Unterscheidung der Wandhälften ergibt sich dann ja bereits aus dem explizit gesetztem *Slab.RotX*. Alternativ dazu könnte man auch andere *PartType*-Werte verwenden, wie etwa „P01“.

3.7.5 Various Slab Information

Sammlung diverser Slab-Informationen. Entspricht im Einzelnen den entsprechenden Feldern im Info-Block der UNICAM-SLABDATE.

Einige dieser Informationen sind redundant, da sie sich aus anderen Größen berechnen lassen. Diese Felder werden hauptsächlich aus UNICAM-Kompatibilitätsgründen mitgeführt. Das gilt im besonderen Maße für die Felder **MaxLength**, **MaxWidth**, **IronProjectionLeft**, **IronProjectionRight**, **IronProjectionBottom**, **IronProjectionTop**: diese Felder sind uneingeschränkt redundant und darüber hinaus nicht drehinvariant (d.h. sie müssten bei einer Drehung des Elementes angepasst werden). Von der Verwendung dieser Felder wird daher generell abgeraten; stattdessen sollten stets die tiefer liegenden geometrischen Detail-Informationen verwendet werden, um diese Werte zu ermitteln.

ReforcemInfo gibt es ebenfalls nur für die Kompatibilität zu UNICAM; es enthält die Angaben des UNICAM-REFORCEM Info-Blocks

Die **Generic Slab Info** Felder stehen für frei verwendbare Zusatzinformationen zur Verfügung.

3.7.6 Multi-Layer Elements

In einer Slab können mehrere Betonschichten angegeben werden (Lots). Ein Elementteil könnte aber auch aus mehreren Schichten bestehen, von denen jeder ihre eigene Bewehrung und eigene Einbauteile hat (Entsprechend den UNICAM-LAYERS).

In PXML kann man das realisieren, indem in den *Outline*- und in den *Steel*-Blöcken die Layer-Werte gesetzt werden.

3.7.7 Legacy Slab Fields

Die hier gelisteten Felder waren ursprünglich auf *Slab*-Ebene definiert worden (aus historischen Gründen die im Wesentlichen der Kompatibilität zu UNICAM entstammen).

Inhaltlich führen diese Felder aber Informationen, die auf *Product*-Ebene liegen, da es sich um Werte handelt, die das ganze Element betreffen (und nicht nur ein Elementteil).

Mit der Einführung der *PXML-Delegate-Files* wurde es unumgänglich, diese Felder auf die Hierarchiestufe zu bringen, der sie inhaltlich angehören, also der *Product*-Ebene. D.h. einige *Slab*-Felder wurden wie folgt durch *Product*-Felder ersetzt:

- *Slab.SlabNo* → *Product.ElementNo*
- *Slab.NumberOfMeansOfTransport* → *Product.StackNo*
- *Slab.TransportSequence* → *Product.StackingSequence*
- *Slab.PileLevel* → *Product.StackingLevel*

- *Slab.PositionInPileX* → *Product.StackingX*
- *Slab.PositionInPileY* → *Product.StackingY*
- *Slab.PositionInPileZ* → *Product.StackingZ*
- *Slab.AngleInPile* → *Product.StackingAngle*

- *Slab.ProjectCoordinates* → *Product.P1X/P1Y/P1Z/P2X/P2Y/P2Z/P3X/P3Y/P3Z*

- *Slab.ProductAddition* → *Product.AdditionInfo*
- *Slab.TypeOfUnloading* → *Product.UnloadingInfo*
- *Slab.MeansOfTransport* → *Product.TransportInfo*
- *Slab.OrderPosition,*
  Slab.ProductGroup,
  Slab.SlabType,
  Slab.ItemDesignation, → *Product.ItemPosition*

Die dadurch obsolet gewordenen *Slab*-Felder werden im Folgenden als **Legacy Slab Fields** bezeichnet. Konzeptionell sind diese Felder verzichtbar, aber aus Gründen der Rückwärtskompatibilität kommt ihnen noch eine gewisse Bedeutung zu. Im Einzelnen wird folgende Rückwärtskompatibilitäts-Strategie empfohlen:

1) In applikationsinternen Datenstrukturen sollten die *Legacy Slab Fields* nicht mehr verwendet werden.
2) Beim Lesen von PXML-Dateien sollten die neuen *Product*-Felder genommen werden, wenn sie vorhanden sind. Für nicht angegebene *Product*-Felder sollte auf die *Legacy Slab Fields* zurückgegriffen werden (wenn mehrere Slabs vorhanden sind, ist erste relevant, wo das betreffende Feld gesetzt ist).
3) Beim Schreiben von PXML-Dateien kann man die *Legacy Slab Fields* weglassen, wenn sichergestellt ist, dass der lesende Prozess die neuen *Product*-Felder kennt. Im Zweifelsfall

sollte aber auf maximale Kompatibilität geachtet werden und die *Legacy Slab Fields* ebenfalls geschrieben werden[21].

3.7.8 Darstellung in vereinfachter Geometrie

Viele ältere Systeme berücksichtigen folgende Felder nicht:

- *Slab.RotX/RotY/RotZ*
- *Slab.ProdX/ProdY/ProdZ*
- *Slab.ProdRotX/ProdRotY/ProdRotZ*
- *Outline.RotX/RotY/RotZ*
- *Steel.RotX/RotY/RotZ*
- *Steel.ProdX/ProdY/ProdZ*

In einer vereinfachten geometrischen Darstellung werden diese Felder auf 0 gesetzt um die Kompatibilität zu solchen älteren Systemen herzustellen. So weit wie möglich werden die Werte dieser Felder hierbei in andere Felder eingerechnet, um den Informationsverlust gering zu halten[22].

3.8 Outline

Unter **Outline** wird hier eine allgemeine geometrische Umrandung verstanden. Das Attribut **Type** bestimmt die Art der Umrandung:

- **lot**
- **mountpart**

Einige Tags des Outline-Abschnittes sind nur für bestimmte Outline-Typen vorgesehen, und werden bei den anderen Outline-Typen ignoriert.

Die **lot**-Outlines beschreiben **Beton-Parzellen** mit Konturen, Aussparungen und Betondaten.

Die **mountpart**-Outlines beschreiben Einbauteile.

3.8.1 Geometric Outline Placement (X, Y, Z, RotX, RotY, RotZ)

Über Verschiebung und Drehung können die einzelnen *Outline*-Objekte innerhalb einer *Slab* platziert werden. Die Operationen werden in folgender Reihenfolge durchgeführt[23]:

1) Verschiebung (X, Y, Z)
2) Drehung RotZ um absolute Z-Achse
3) Drehung RotY um absolute Y-Achse

---

[21] Theoretisch würde es sogar ausreichen, nur die *Legacy Slab Fields* zu schreiben, also ohne die entsprechenden neuen *Product*-Felder anzugeben. Die Datenübergabe an ERP-Systeme würde dadurch aber deutlich erschwert, da das ERP-System bis zur *Slab*-Ebene hinuntergehen müsste. Und spätestens bei Verwendung von *PXML-Delegate-Files* ist es unumgänglich, die neuen *Product*-Level-Felder zu setzen.

[22] Der größte Informationsverlust entsteht hierbei beim Drehen von *Outline*- oder *Girder*-Objekten um die X- oder Y-Achse. Hier kann empfohlen werden, die Operation auf die x- und y-Komponenten zu beschränken. Konkret bedeutet das: wenn um die X-Achse um den Winkel φ gedreht wird, wird folgende Transformation durchgeführt:

$$Y \coloneqq cos\varphi \cdot Y$$

$$\alpha \coloneqq arctan2(cos\varphi \cdot sin\alpha, cos\alpha)$$

Hierbei steht *Y* für alle Y-Koordinaten, die auf *SVertex.Y* wirken und α steht für *Girder.AngleToX*.
Bei der Drehung um die Y-Achse um den Winkel φ hat man eine analoge Berechnung:

$$X \coloneqq cos\varphi \cdot X$$

$$\alpha \coloneqq arctan2(sin\alpha, cos\varphi \cdot cos\alpha)$$

Eine 180°-Drehung um die X- bzw. Y-Achse wird so zu einer Spiegelung um die XZ- bzw. YZ-Ebene.

[23] Die Reihenfolge der Operationen ist exakt einzuhalten. Die gewählte Festlegung mag etwas ungewöhnlich wirken und durch Kompatibilitätsüberlegungen motiviert.

4) Drehung RotX um absolute X-Achse

Die Verschiebe-Offsets sind in mm und die Drehwinkel sind in DEG.

Einbauposition

In UNICAM gibt es (ab Version 6.0) für Einbauteile die Angabe einer **Einbauposition**, bezogen auf den *Slab*-Nullpunkt. Dieser Wert ist bis zu einem gewissen Grad redundant, da die Einbauposition aus der Position und Geometrie des Einbauteils folgt. (Das gilt zwar nur für bekannte Einbauteile, doch können auch nur solche automatisch verlegt werden). Aus diesem Grund gibt es in PXML diese Positionsangabe nicht.

Man kann aber den X/Y-Offset des Einbauteils so setzen, dass er mit der Einbauposition übereinstimmt (die Vertex-Koordinaten müssen entsprechend kompensiert werden). Es wird empfohlen, dies beim Datenaustausch mit UNICAM so zu handhaben.

3.8.2 Height

Höhe in mm (Dicke der Betonschicht, Höhe des Einbauteils).

3.8.3 Name

Eine Bezeichnung des Lots. Bei der Übernahme von UNICAM-Contour-Schichten wird hier das Kennzeichen der Schicht eingetragen. Bei der Übernahme von UNICAM Mountparts wird die Mountpart-Bezeichnung eingetragen.

3.8.4 GenericInfo

Frei verwendbare Informationszeilen.

3.8.5 MountingInstruction (only for mountparts)

Einbau-Anweisung. In Anlehnung an das UNICAM-Einbau-Kennzeichen gilt:

- 0 = Teil wird eingebaut
- 1 = Teil wird nur gezeichnet
- 2 = Teil wird nur eingebaut
- 3 = Teil wird weder gezeichnet noch eingebaut
- 4 = Teil wird in Bewehrung eingebaut
- 5 = Teil wird automatisch eingebaut

3.8.6 MountPartType, MountPartArticle (only for mountparts)

Entspricht den entsprechenden UNICAM-Definitionen.

In Erweiterung dazu wird der *MountPartType* „**21**“ allgemeiner als **Abzugskörper** definiert. Einbauteile dieses Typs können verwendet werden, um Aussparungen und Hohlräume im Beton zu modellieren. Aussparungen können demnach alternativ über Löcher in der Geometrie modelliert werden (cutout-Shape), oder über Abzugskörper (Die Modellierung über Abzugskörper ist zwar etwas komplexer, dafür aber weit flexibler)[24].

3.8.7 MountPart Properties (only for mountparts)

Entspricht den entsprechenden UNICAM-Definitionen:

- **MountPartIronProjection** = Eisenüberstand in mm. Es handelt sich hierbei um Eisen, die aus dem Einbauteil herausragen.

[24] Beachte: ein Abzugskörper muss in seiner MountingInstruction immer als einzubauendes Teil gekennzeichnet werden, da ein nicht einzubauendes Teil kein Volumen verdrängt.

- **MountPartDirection** = Richtung des Eisenüberstandes oder Ausrichtung des Einbauteils. Die Winkelangabe ist im Bereich]-180°, 180°]. Beim Export nach UNICAM wird dies auf [0°, 360°] konvertiert.
- **MountPartLength/MountPartWidth** = Länge/Breite in mm.
  Die Werte sind optional, und werden im allgemeinen gesetzt, um das automatische Zuschneiden des Teils zu ermöglichen. In diesem Sinne handelt es sich um Produktionsmaße des Einbauteils.
  Benötigt werden die Werte grundsätzlich nur für Einbauteile, deren Größe variabel ist, und nicht durch die Artikelnummer vollständig bestimmt ist.
  Die genaue Bedeutung der beiden Maße hängt vom Typ des Einbauteils ab. Es kann Typen geben, die nur eines der beiden Maße benötigen, und es ist auch möglich, den beiden Werten eine andere Bedeutung, als jene der Länge und Breite, zuzuordnen.

3.8.8 Concrete Properties (only for lots)

- **ConcretingMode** = Betonierkennzeichen.
- **ConcreteQuality** = Betongüte (z.B. B25). Hier können zusätzlich zur eigentlichen Betongüte auch weitere Informationen anlagenspezifisch angegeben werden, so z.B. „B25.Red“ für einen farbigen Beton.
- **UnitWeight** = Rohdichte in kg/dm³.
- **Volume** = Soll-Betonvolumen m³.

3.8.9 Layer

Definiert bei mehrschichtigen Elementen die Layer-Zugehörigkeit.

3.8.10 ObjectID

Die *ObjectID* ermöglicht die Identifikation des Objektes mittels einer ID, die typischerweise vom CAD gesetzt wird. Die *ObjectID* ist im Gegensatz zur *GlobalID* nicht notwendigerweise eindeutig; denn es ist möglich, über diese ID mehrere *Outline*- und/oder *Steel*- Objekte zu verbinden, indem über eine gemeinsame *ObjektID* ihre Zugehörigkeit zu einem gemeinsamen zugrundeliegenden Objekt deklariert wird.

Die *ObjectID* kann auch mehrstufig aufgebaut sein, indem Teil-IDs durch Punkte getrennt werden. Dadurch ist es möglich, komplexe mehrstufige Objekt-Hierarchien aufzubauen.

Beispiel:

Wir betrachten folgende Objekte:

A) ObjectID = „5“
B) ObjectID = „23“
C) ObjectID = „23.1“
D) ObjectID = „23.1.1“
E) ObjectID = „23.1.2“
F) ObjectID = „23.1.2“

Das Objekt A steht für sich allein, während die Objekte B, C, D, E und F zusammengehören, da sie die Gruppe „23“ bilden.

Die Objekte C, D, E und F bilden zudem die Gruppe „23.1“, die eine Untergruppe der Gruppe „23“ ist.

Die Objekte E und F bilden die Gruppe „23.1.2“, die eine Untergruppe der der Gruppe „23.1“ ist.

Über die *ObjectID* ist es auch möglich, *Steel*-Objekte mit *Outline*-Objekten zu verbinden, bzw. komplexe Strukturen aufzubauen, in denen sowohl *Steel*- als auch *Outline*-Objekte in beliebiger Verschachtelung vorkommen. Es kann über die *ObjectID* auch eine Verbindung zu den *ElementInfo*-Objekten hergestellt werden.

Grundsätzlich ist es auch möglich, über die *ObjectID* Objekte zu verbinden, die in unterschiedlichen Elementen liegen. Das ist beispielsweise dann sinnvoll, wenn man große Einbauteile hat, die sich über mehrere Elemente erstrecken.

Die *ObjectID* definiert auch eine Teilordnung zwischen den Objekten, in dem Sinne, dass beispielsweise „23“ vor „23.1“ kommt[25]. Über diese Ordnungsrelation kann man nun Prioritäten für gemeinsame Eigenschaften festlegen: wenn man beispielsweise für die Gruppe „23“ einen gemeinsamen Artikelkode verwenden möchte, verweist man auf den Artikelkode von B, weil B das erste Objekt dieser Gruppe ist (im Sinne der genannten Teilordnung).

3.8.11 Shape, SVertex

Ein **Shape** enthält eine Folge von Punkten (**SVertex**), die ein Polygon bilden. Das Polygon wird geschlossen, indem der letzte Punkt wieder mit dem ersten Punkt verbunden wird (*ohne* dass der erste Punkt deswegen zweimal angeführt würde).

Cutout:

Wenn dieses Feld auf *true* steht, beschreibt das *Shape*-Objekt eine Aussparung, also ein Loch in einer umhüllenden Kontur.

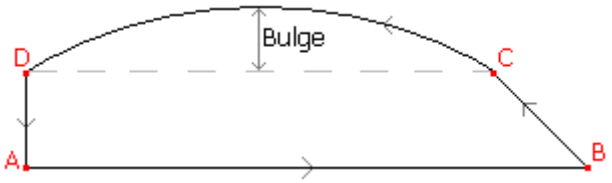
RefHeight:

Eine Referenz-Z-Position, die für die Definition von schrägen Kanten benötigt wird. Siehe unten.

SVertex:

Ein SVertex beschreibt einen Eckpunkt des Polygons (Punkt in der Ebene).

Jedem *SVertex* kann eine Ausbuchtungshöhe **Bulge** zugeordnet werden: falls diese ungleich 0 ist, ist die Verbindung des Vertex zu seinem Folge-Vertex ein Bogen mit der angegebenen Höhe (in mm).



Die Abbildung zeigt ein *Shape* mit 4 *Vertices*. A, B und D habe *Bulge=0*, während für C der Wert von *Bulge* positiv ist; bei einem negativen Wert von *Bulge* wäre der Bogen nach innen gewölbt, d.h. das Vorzeichen von *Bulge* gibt den Bogen-Umlaufsinn an.

Sonderfall von nur einem SVertex:

Falls der *Shape* aus einem einzigen *Vertex* besteht, wird dies als *Kreis* interpretiert. Der *Vertex* gibt den Kreismittelpunkt an und der Betrag von *Bulge* entspricht dem Kreisdurchmesser (dieser kann auch 0 sein); das Vorzeichen von *Bulge* gibt den Umlaufsinn an.

Line-Attribute:

Für jeden *SVertex* kann ein **LineAttribute** angegeben werden, dem man verschiedenste Applikationsspezifische Bedeutungen zuordnen kann. So könnte beispielsweise über das *LineAttribute* einer Elementkontur die Information über den Schalungstyp angegeben werden.

[25] Es ist nur eine Teilordnung eindeutig definiert, nicht eine Totalordnung. Denn während zwischen B und C eine eindeutige Ordnungsrelation besteht, ist die Relation zwischen E und F nicht mehr ohne zusätzliche Bedingungen definierbar. Und selbst die Relation zwischen A und B kann nur bestimmt werden, wenn man festlegt, ob man eine rein alphabetische Sortierung oder eine sogenannte natürliche Sortierung anwendet.

Meistens wird für das *LineAttribute* eine 4-stellige Hex-Zahl angegeben, die als Bit-Feld interpretiert wird. Die einzelnen Bits haben dann folgende Bedeutung:

Bit 00 [0001]: Keine Fase unten.
Bit 01 [0002]: Sonderschalung.
Bit 02 [0004]: Vergussfuge.
Bit 03 [0008]: Keine Fase oben.
Bit 04 [0010]: Rundung unten.
Bit 05 [0020]: Feder (Schalung mit Nut).
Bit 06 [0040]: Nut (Schalung mit Feder).
Bit 07 [0080]: Rundung oben.
Bit 08 [0100]: Saubere Kante.
Bit 09 [0200]:
Bit 10 [0400]:
Bit 11 [0800]: Stützschalung (Schalung stützt ein Einbauteil).
Bit 12 [1000]: Fensterschalung.
Bit 13 [2000]: Konturschalung.
Bit 14 [4000]: Schalung nicht fixieren.
Bit 15 [8000]:

Offener Polygonzug:

Nach obiger Definition bilden die *Vertices* eines *Shapes* immer einen geschlossenen Polygonzug. Ein offener Polygonzug kann nur dadurch realisiert werden, dass man vom letzten Punkt wieder den gesamten Weg bis zum ersten Punkt zurückläuft. Insbesondere wird ein *offener Kreisbogen* durch ein Shape mit 2 *Vertices A* und *B* realisiert, die $A_{Bulge}=-B_{Bulge}$ erfüllen. Falls die beiden *Bulge*-Werte zudem gleich *0* sind, hat man den Spezialfall einer einfachen *Strecke*.

Beachte: offene Polygonzüge (vereinzelte Strecken oder Bögen) haben immer einen Flächeninhalt von 0. Sie sind daher nur beim *mountpart*-Outlines sinnvoll; bei *lot*-Outlines würden offene Polygonzüge als Konturen und Aussparungen mit Fläche 0 interpretiert, was offenbar sinnlos ist. Eine Kontur oder Aussparung darf daher nie als Folge von Einzelstrecken dargestellt werden, sondern muss stets über Polygone dargestellt werden.

Kanten mit Profil:

Das **Profile**-Feld ermöglicht eine der exakten Festlegung des geometrischen Profils der Kanten. Das Feld enthält eine Zeichenkette der Form

$$z_0|p_0\ z_1|p_1\ z_2|p_2\ \ldots\ z_n|p_n$$

Es wird also eine Folge von Zahlenpaaren angegeben, die durch Leerzeichen getrennt sind (die Zahlen selbst sind beliebige XML-Fließkommawerte)[26].

Die z_i sind vertikale Werte (Z-Werte) und die p_i sind horizontale Werte, wobei positive Horizontalwerte aus dem Beton herausragen[27]. Es sei hier ausdrücklich darauf hingewiesen, dass das

[26] Die Angabe des ersten und des letzten Z-Wertes ist optional: lässt man der ersten Z-Wert weg, wird er zu 0 angenommen; lässt man den letzten Z-Wert weg, wird für ihn die Dicke der Betonschicht plus *Outline*-Z-Offset angenommen. Wenn beide Z-Werte weggelassen werden, hat der Profilstring folgendes Format:

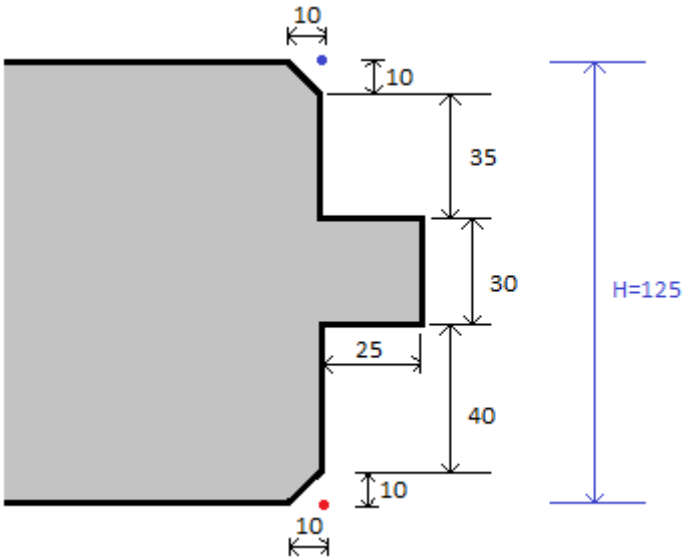
```
p0 z1|p1 z2|p2 … zn-1|pn-1 pn
```

Dieses verkürzte Format wird aber nur aus Gründen der Rückwärtskompatibilität unterstützt. Von der Verwendung dieser Kurzform wird abgeraten.

[27] Die Festlegung, dass positive Horizontalwerte aus dem Beton herausragen, setzt voraus, dass immer eindeutig zwischen „innen“ und „außen“ unterschieden werden kann. Das ist bei Polygonen mit nichtverschwindender Fläche

Profile-Feld die Beton-Form beschreibt; bei Aussparungen und Einbauteilen (Schalungen) sind die Werte gespiegelt zu interpretieren, d.h. ein negativer `pi`-Wert ragt aus dem Einbauteil heraus.

Beispiel:

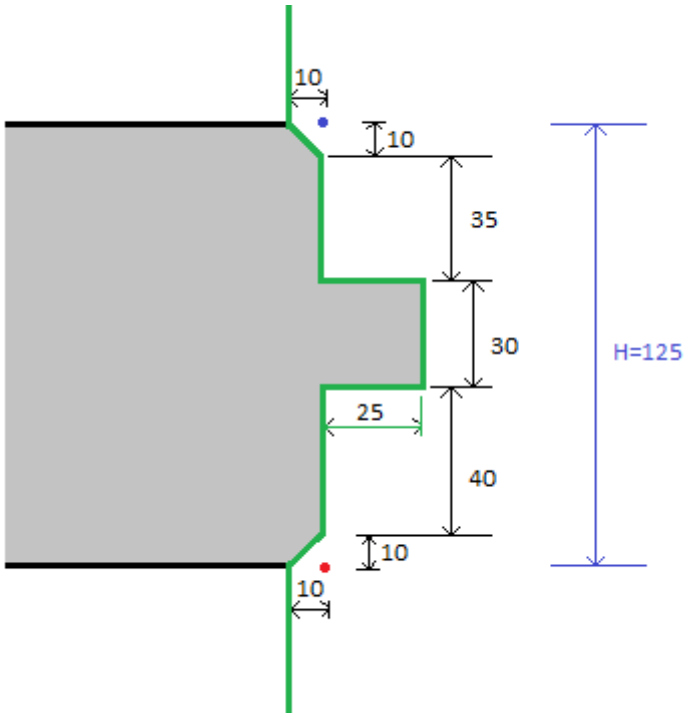


Dieses Betonprofile wird durch folgenden *Profile*-Eintrag beschrieben:

```
<Profile>0|-10 10|0 50|0 50|25 80|25 80|0 115|0 125|-10</Profile>
```

Der rote und der blaue Punkt in der Zeichnung beschreiben den unteren und den oberen Punkt der nominalen Kante.

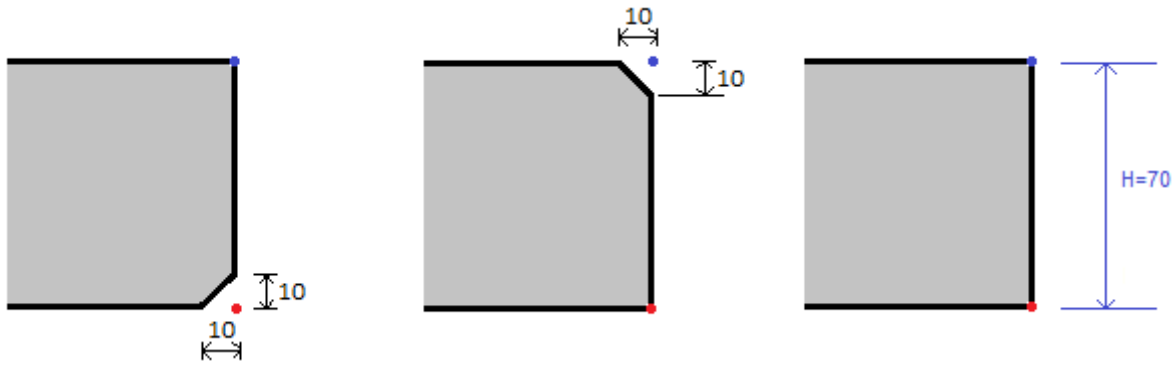
Das so beschriebene Profil wird nach unten und oben ins unendliche verlängert, so dass sich ein für alle z-Werte definiertes Profil ergibt (siehe grüne Linie):



Es ergeben sich folgende wichtige Beispiele für „Fase unten“, „Fase oben“ und „keine Fase“:

---

immer möglich. Bei Polygonen mit weniger als 3 Punkten ist das aber nicht möglich, da ein Polygon mit Fläche 0 keinen festgelegten Umlaufsinn hat. Für solche Fälle muss man einfach den Umlaufsinn willkürlich festlegen, und zwar positiv für alle Betonumrandungen und negativ für die Aussparungen und Einbauteile.



`<Profile>60|0 70|-10</Profile>`

`<Profile>0|-10 10|0</Profile>` `<Profile>0|0</Profile>`

Die z_i beziehen sich auf einen „absoluten“ Nullpunkt, der bei *–Outline.Z* angenommen wird. Dieses Detail ist aber nur für mehrschichtigen *Slabs* relevant (siehe unten).

Der Profilstring ist definitionsgemäß nie leer. Wenn also im *Profile*-Feld ein leerer String steht, bedeutet das, dass kein *Profile* angegeben wurde.

Das *Profile*-Feld gehört zwar einem bestimmten *SVertex* an, bezieht sich aber, so wie auch das *LineAttribute*, auf die gesamte auf *SVertex* folgende Kante.

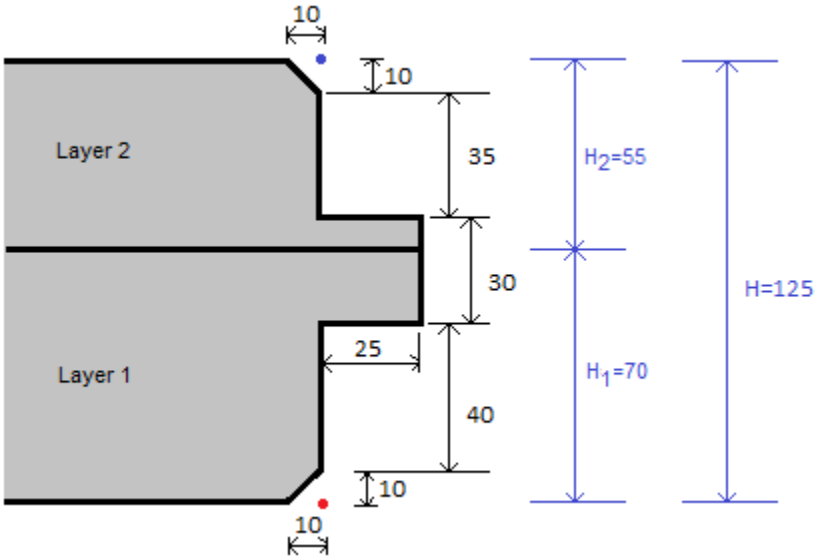
Das *Profile*-Feld ermöglicht die exakte geometrische Beschreibung der Betonkontur. Damit entsteht eine gewisse Redundanz zu *LinienAttribute*, denn auch über *LinienAttribute* können Konturinformationen wie Fase, Feder und Nut angegeben werden. Das *LinienAttribute* ist aber eher im Sinne einer Produktionsdirektive zu verstehen: es gibt an, wie produktionstechnisch erreicht werden soll, dass die gewünschte Geometrie entsteht und die passenden Schalungssysteme zum Einsatz kommen. Am Ende obliegt es aber der Maschinenlogik, zu entscheiden, ob die Maschine den *LinienAttribute* -Direktiven folgt, oder ob sie sich nur auf die Geometrieinformationen stützt.

Profil bei mehrlagigen Slabs:

Bei mehrschichtigen *Slabs* (Sandwich-Elementen) betrachtet man ein *Profil*, das den gesamten Z-Bereich der *Slab* umfasst und somit über die einzelne Betonschicht hinausragt. Für jede Schicht ist dann nur jener Abschnitt des *Profils* relevant, der im Z-Bereich der jeweiligen Schicht liegt.

Um für übereinanderliegenden Betonschichten exakt den selben *Profil*-Code verwenden zu können, definieren wir, dass sich die Z-Werte des *Profile*-Strings auf eine absolute Z-Lage beziehen. Um lokale Z-Werte zu erhalten (die auf den *Outline*-Nullpunkt bezogen sind), muss man folglich *Outline.Z* von allen z_i subtrahieren.

Beispiel:



Beide Betonschichten sollten hier das selbe *Profil* haben, nämlich

```
<Profile>0|-10 10|0 50|0 50|25 80|25 80|0 115|0 125|-10</Profile>
```

Falls eine mehrlagige *Slab* auch Isolierschichten enthält, müssen auch diese Isolierschichten zum Gesamtprofil passen. Hier müssen 2 Fälle unterschieden werden:

a) Isolierungen die als Einbauteile eingelegt werden: Diese Isolierungen werden in einem getrennte Produktionsschritt hergestellt. Ihre Form wird nicht durch die Betonschalung gebildet. Diese Isolierungen sind als *mountpart-Outlines* darzustellen und haben nicht den selben *Pofil*-Code wie die darunter oder darüber liegenden Betonschichten (Der *Profil*-Code würde bei Einbauteilen auch invertiert interpretiert).

b) Isolierungen die flüssig oder schaumförmig ausgetragen werden und folglich von der Betonschalung geformt werden: Diese Isolierungen sind als Betonschichten darzustellen, also als *lot-Outlines*. Sie tragen dann die selben *Profil*-Code wie die darunter oder darüber liegenden Betonschichten.

Schräge Kanten (DX, DY, RefHeight)[28]*:*

Wenn ein Objekt schräge Kanten hat, so beschreiben die Felder *X* und *Y* die Vertex-Koordinaten auf der Z-Höhe **Shape.RefHeight** (gemessen von der Z-Lage des Objekts). Typischerweise ist *RefHeight* = 0; dann beziehen sich X und Y auf die Z-Basislage des Objektes.

Auf allen anderen Z-Lagen ist die X/Y-Position gegeben durch

$$X + DX * (Z - RefHeight)$$

$$Y + DY * (Z - RefHeight)$$

Falls *DX* und/oder *DY* ungleich 0 sind, verläuft die *Outline*-Kante nicht parallel zur Z-Achse nach oben, sondern ist schräg. Bei einem *DX*-Wert von 1 (und *DY* = 0) hat die Kante beispielsweise einen 45°-Winkel zur Z-Achse.

Es können solcherart beispielsweise schiefe Prismen und Pyramiden realisiert werden. Um allgemeinere Polyeder realisieren zu können, muss man eventuell mehrere *Outline*-Objekte zusammensetzen.

[28] Hier wird die Darstellung von schrägen Kanten mittels der Felder *DX* und *DY* beschrieben. Für die Festlegung der seitlichen Betonkontur ist die *Profile*-Darstellung aber einfacher und leistungsfähiger als die Beschreibung über *DX/DY*.

Man beachte, dass sich *DX* und *DY* auf eine *SVertex* beziehen; eine Kante wird daher durch beide angrenzenden Vertices beeinflusst. In diesem Punkt unterscheidet sich das *DY*/*DY*-Konzept wesentlich vom *Profile*-Konzept, bei dem eine Kante immer nur vom vorhergehenden *Profile*-Eintrag bestimmt wird[29].

Anmerkung: Wenn zwei benachbarte Vertex-Kanten nicht in einer gemeinsamen Ebene liegen (also windschief zueinander stehen), können diese nicht über eine ebene Fläche verbunden werden. In einem solchen Fall stellt sich natürlich die Frage, wie die Verbindungsfläche überhaupt zu definieren ist.

Für eine grafische 3D-Darstellung ist es naheliegend, die Verbindungsfläche einfach durch 2 Dreiecke darzustellen – das ist der einfachste Weg, zumal 3D-Grafiken ohnedies typischerweise aus Dreiecken zusammengesetzt sind.

Um eine einheitliche Volumenberechnung zu ermöglichen, benötigt man aber eine bindende Festlegung. Die PXML-Empfehlung hierzu lautet, den Körper als eine Aufeinanderfolge von gerader Prismen mit infinitesimaler Höhe aufzufassen. Die Volumenberechnung ist dann wie folgt gegeben:

$$V = \frac{Height}{12}\sum_{i=0}^{n-1}\left(\left(6X_{L,i} + 3X_{H,i}\right)Y_{L,i+1} - \left(6Y_{L,i} + 3Y_{H,i}\right)X_{L,i+1} + \left(3X_{L,i} + 2X_{H,i}\right)Y_{H,i+1} - \left(3Y_{L,i} + 2Y_{H,i}\right)X_{H,i+1}\right)$$

Hierbei sind X_L, Y_L die Koordinaten am unteren Ende des Objektes, und X_H, Y_H die Koordinaten am oberen Ende des Objektes. Der Index i zählt die Polygonpunkte durch; *Height* ist die Gesamthöhe des Objektes.

Vereinfachte Volumenberechnung: Die oben angeführte exakte Volumenberechnung lässt sich für isolierte Körper gut ausführen. Wenn man aber schräge Körper mit Löcher hat oder schräge Einbauteile, die teilweise in Betonlots hineinreichen, wird die Komplexität der exakten Berechnung sehr hoch. Für die meisten Anwendungen wird es daher sinnvoller sein, eine vereinfachte Volumenberechnung zu durchzuführen, indem alle DX- und DY-Werte zu 0 angenommen werden.

3.9 Steel

Der *Steel*-Abschnitt ist eine Gruppierung von Rundeisen und Gitterträgern.

Type-Attribut

Es gibt folgende Typen:

- **none:** Nicht geheftete Bewehrung, typischerweise lose Eisen und Gitterträger ohne Zusammenbau-Anweisungen und häufig auch ohne Angabe einer räumlichen Anordnung.
- **mesh:** Geheftete oder verschweißte Bewehrung mit verpflichtender Angabe der genauen räumlichen Anordnung der Eisen und Gitterträger.
- **cage:** Gleichbedeutend mit *mesh*. Die Unterscheidung *mesh*/*cage* ist nur historisch bedingt und hat höchstens anlagenspezifische Bedeutung.
- **extiron:** Getrennt zugeführte Bewehrung, die nicht im Detail spezifiziert wird. Das können neben losen Eisen auch komplexere Einheiten wie Matten oder Körbe sein. Der Gitterträger-Teil wird bei Extiron-Steelblöcken nicht verwendet.

Zusammenfassend können wir sagen, dass der Unterschied zwischen *"none"* und *"mesh/cage"* darin besteht, dass wir bei *"none"* nur eine Liste von Stäben haben, während wir bei *"mesh/cage"* zusätzlich einen definierten Montageprozess auf *Steel*-haben. In diesem Fall kann der *MeshType* (siehe unten) weitere Anweisungen für die Verarbeitung auf *Steel*-Ebene geben.

[29] Übrigens stellen *Profile* und *DX*/*DY* alternative Beschreibungsverfahren dar, die schwerlich gemeinsam eingesetzt werden. Theoretisch ist es aber schon möglich beide Angaben zu kombinieren: die Verschiebungen in X und Y von *Profile* und jene von *DX/DY* sind dann zu addieren.

3.9.1 Geometric Steel Placement (X, Y, Z, RotX, RotY, RotZ)

Über Verschiebung und Drehung können die einzelnen *Steel*-Objekte innerhalb einer *Slab* platziert werden. Die Operationen werden in folgender Reihenfolge durchgeführt[30]:

1) Verschiebung (X, Y, Z)
2) Drehung RotZ um absolute Z-Achse
3) Drehung RotY um absolute Y-Achse
4) Drehung RotX um absolute X-Achse

Die Verschiebe-Offsets sind in mm und die Drehwinkel sind in DEG.

3.9.2 ToTurn (Nur für Matten)

Mögliche Werte: **true**, **false**.

Gibt an ob die Matte gewendet ausgeliefert werden soll (gewendet wird entlang der Längsachse).

3.9.3 StopOnTurningSide (Nur für Matten)

Mögliche Werte: **true**, **false**.

Gibt an ob die Anschlagsseite der Matte zugleich auch die Wendeseite ist.

3.9.4 Name

Bezeichnung des Steel-Blocks (z.B. Mattenbezeichnung).

3.9.5 MeshType

Bei Steel-Blöcken vom Typ *mesh*/*cage* ist prinzipiell eine maschinelle Herstellung der gesamten Einheit möglich. Es kann dann über den **MeshType** ein Produktionsverfahren vorgegeben werden:

- 0: Standardverfahren.
- 1: Mattenbiegen 2D: Herstellung über Balkenbiegemaschine.
- 2: Manuelle oder teilautomatisierte Herstellung eines Bewehrungsmoduls (z.B. durch manuelles Verscheißen automatisch positionierter Bügelserien).
- 3: Mattenbiegen 3D: Herstellung über Einzelkopfbiegemaschine.
- 4: Deckelmatte (z.B. Deckel eines Korbes, oder einer Massivwand-Bewehrung).
- 5: Deckelmatte 2D; wie Typ '1', wird aber zusammen mit Typ '4' ausgeliefert.
- 6: Deckelmatte 3D; wie Typ '3', wird aber zusammen mit Typ '4' ausgeliefert.
- 7:
- 8: Lose Matte: wird nicht von der Stahlmaschine produziert, sondern manuell hinzugefügt (üblicherweise wird eine solche Matte einem Matten-Lager entnommen).
- 9: Applikationsspezifischer Typ.

Der *MeshType* kann neben der oben genannten Typkennung noch weitere typspezifische Detail-Informationen beinhalten; Folgende Verwendung wird empfohlen:

- 8#StandardSheet:R188A
  Lose Matte, zu realisieren mittels Lagermatte vom Typ R188A.
- 9#ProgressInfo8080:123#ProgressInfo8090:abc
  Hier wird ein applikationsspezifischer Typ angegeben, der über 2 applikationsspezifische

[30] Die Reihenfolge der Operationen ist exakt einzuhalten. Die gewählte Festlegung mag etwas ungewöhnlich wirken und durch Kompatibilitätsüberlegungen motiviert.

Parameter näher beschrieben wird (die Parameter haben in diesem Beispiel die Bezeichnung "ProgressInfo8080" und "ProgressInfo8090"; diesen Parametern werden die Werte "123" bzw. "abc" zugewiesen).

Es können also mehrere Parameter angegeben werden, die durch #-Zeichen getrennt sind; Parametername und der Parameterwert sind durch einen Doppelpunkt getrennt.

Anmerkung: Der *MeshType* selbst beschreibt das grundlegende Herstellungsverfahren, ohne aber im Detail festzulegen, wie die einzelnen Eisen zu verarbeiten sind. Die Rolle der einzelnen Eisen innerhalb des festgelegten Herstellungsverfahrens wird dann erst durch den *ReinforcementType* der Eisen festgelegt (siehe Abschnitt 3.10.2).

3.9.6 WeldingDensity (Nur für Matten)

Schweißdichte in % (Integer-Wert).

Anzugeben sind Werte zwischen 0 und 100. Zusätzlich können höherwertige Stellen für die Kodierung von Zusatzinformationen verwendet werden:

a = WeldingDensity **mod** *1000*

b = WeldingDensity **div** *1000.*

Der Wert von *a* bestimmt die Schweißdichte; der Wert von *b* steht für zusätzliche anlagenspezifische Informationen zur Verfügung.

Falls *WeldingDensity* einen Wert von *0* oder *DBNull* hat, so steht dies für "Default"; je nach Anlage kann dies 0%, 100% oder irgendein anderer Wert sein.

Beachte: die angegebene Schweißdichte ist als *innere* Schweißdichte zu verstehen; am Mattenrand werden *zusätzliche* Schweißpunkte eingefügt (siehe *BorderStrength*).

3.9.7 BorderStrength

Verstärkung des Mattenrandes. Ein Wert von *0* oder *DBNull* steht für den anlagenspezifischen Default-Wert. Ein Wert von 1 bedeutet, dass die äußerste Reihe zu 100% geschweißt wird; ein Wert von 2 bedeutet, dass die 2 äußersten Reihen zu 100% geschweißt werden.

3.9.8 Generic Steel Info

Frei verwendbare Infozeilen.

3.9.9 Steel Production Directives (ProdX/Y/Z, ProdRotX/Y/Z)

Die *Steel Production Directives* Felder beschreiben eine produktionstechnische Positionierung des Bewehrungs-Blocks (typischerweise eines Korbes), der aber nicht das fertige Produkt betrifft, sondern nur als Empfehlung für die Bewehrungsproduktionsmaschine zu verstehen ist. Der Bewehrungskorb ist also *nicht* in gedrehter oder verschoben Form in das Betonelement einzubauen, sondern nur vorübergehend so zu positionieren, dass er leichter gefertigt werden kann. Der Bewehrungsproduktionsmaschine steht es frei, diese Empfehlung zu beachten, oder eigenständig zu entscheiden, wie der Korb positioniert werden soll, um gut gefertigt werden zu können[31].

Die ProdRot-Winkel sind in DEG-Einheit anzugeben, und die Reihenfolge der Operationen ist wie folgt:

1) Drehung *ProdRotZ* um absolute Z-Achse
2) Drehung *ProdRotY* um absolute Y-Achse
3) Drehung *ProdRotX* um absolute X-Achse
4) Verschiebung um *ProdX/Y/Z*

[31] Beachte: Üblicherweise betrifft die Drehung nur die Rundstahl-Eisen, nicht aber die Gitterträger.

Die am Ende resultierende Produktionslage der Bewehrung ergibt sich aus der Aufeinanderfolge aller Dreh- und Verschiebungsoperationen:

1) *Steel.X/Y/Z*
2) *Steel.RotZ*
3) *Steel.RotY*
4) *Slab.RotX*
5) *Slab.X/Y/Z*
6) *Slab.RotZ*
7) *Slab.RotY*
8) *Slab.RotX*
9) *Slab.ProdRotX*
10) *Slab.ProdRotY*
11) *Slab.ProdRotZ*
12) *Slab.ProdX/ProdY/ProdZ*
13) *Steel.ProdRotZ*
14) *Steel.ProdRotY*
15) *Steel.ProdRotX*
16) *Steel.ProdX/Y/Z*

Wenn man die jeweils zusammengehörigen Dreh- und Schiebe-Operationen einer Stufe in je eine Drehschiebung *H* zusammenfasst, so hat man folgende Aufeinanderfolge:

1) H_{Steel}
2) H_{Slab}
3) $H_{SlabProd}$ → Belegung
4) $H_{SteelProd}$ → Arrangement

Oder als Operator-Multiplikation geschrieben:

$$H_{total} = H_{SteelProd} \cdot H_{SlabProd} \cdot H_{Slab} \cdot H_{Steel}$$

Oft bezeichnet man $H_{SlabProd}$ als **„Belegung“** (Palettenbelegung, Bahnbelegung) und $H_{SteelProd}$ als **„Arrangement“**.

<u>Prozess-Reihenfolge von Belegung und Arrangement</u>: Im Arbeitsvorbereitungsprozess wird die *Belegung* meistens vor dem *Arrangement* festgelegt. Das passt gut zur oben definierten Operator-Reihenfolgen, da das *Arrangement* sich dann auch eine vorgegebene *Belegung* bezieht. In Szenarien komplexer Bewehrungsproduktion kann es aber auch vorkommen, dass das *Arrangement* bereits optimiert wird, bevor die *Belegung* festgelegt wird. Da sich das *Arrangement* aber auf eine gegebene *Belegung* bezieht, muss beim nachträglichen Verändern der *Belegung* auch das *Arrangement* angepasst werden, um das effektive *Arrangement* faktisch unverändert zu lassen.

Konkret: wenn wir mit $\widehat{H}_{SteelProd}$ das *Arrangement* bezeichnen, das vor dem Setzten der *Belegung* optimiert wurde, so muss man, wenn man nachträglich $H_{SlabProd}$ setzt, $H_{SteelProd}$ anpassen:

$$H_{SteelProd} = H_{SlabProd} \cdot \widehat{H}_{SteelProd} \cdot H_{SlabProd}^{-1}$$

3.9.10 Layer

Definiert bei mehrschichtigen Elementen die Layer-Zugehörigkeit.

3.9.11 ObjectID

Siehe gleichlautenden Abschnitt zu den *Outline*-Objekten, Abschnitt 3.8.10

3.10 Bar

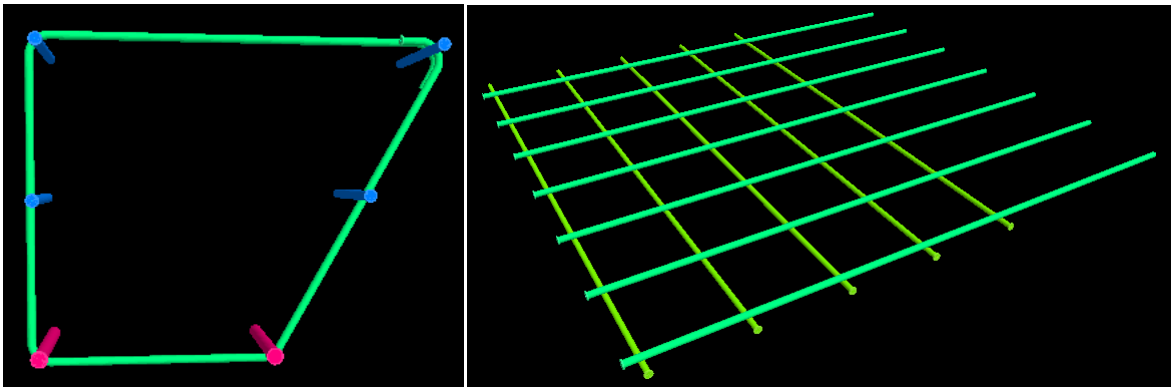
Ein *Bar*-Eintrag entspricht einem Rundstahl-Element, also einem geraden oder gebogenen Bewehrungseisen.

3.10.1 ShapeMode

Über den *ShapeMode* hat man die Möglichkeit eine Darstellungsart für die Geometrie des Eisens zu wählen. Für gerade Eisen spielt diese Option keine große Rolle; für komplexe Biegeformen hat man durch geeignete Wahl des *ShapeMode* aber die Möglichkeit, die Datendarstellung möglichst stark an die interne Darstellung im CAD anzulehnen. Der Aufwand für die Implementierung des Datenexports vom CAD kann dadurch entscheidend verringert werden.

3.10.1.1 ShapeMode "realistic"

Alle Eisen werden mit korrekten Biegeradien und korrekten Raumkoordinaten angegeben (die Biegeradien können auch über die *BendingDevice* angegeben werden). Die Relativ-Abstände der Eisen sind ebenfalls korrekt zu berücksichtigen. Diese Darstellung ist uneingeschränkt verwendbar, da sie das zu erzeugende Produkt direkt wiedergibt, und so keine Interpretationsspielräume lässt.

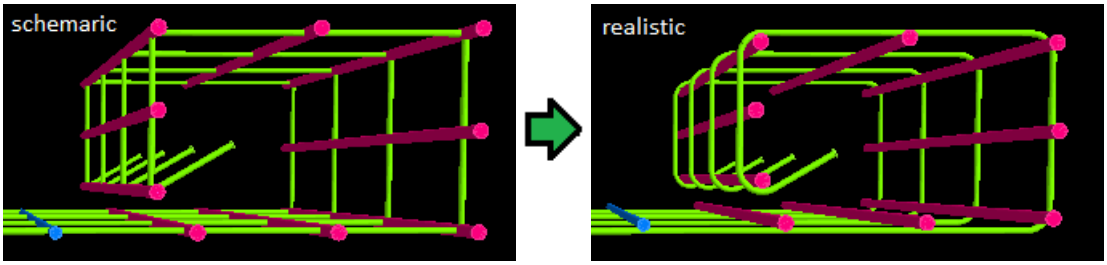


Die Bewehrungsart der Eisen hat in dieser Darstellung keine geometrische Bedeutung, da bereits ohne Kenntnis der Bewehrungsart die Geometrie eindeutig definiert ist.

Wegen ihrer Universalität und Eindeutigkeit ist das die empfohlene Darstellungsart.

3.10.1.2 ShapeMode "schematic"

Der *schematic*-Mode ist eine vereinfachte Darstellung in der die Eisen mit Biegeradius 0 gezeichnet werden:



Bei Biegewinkeln bis zu 90° ist die Biegeform einfach nur "eckiger" als in der Realität. Für Biegewinkel über 90° wird aber die äußere Form verfälscht dargestellt.

Die räumliche Absolut-Lage der Eisen ist hierbei wie folgt zu setzen:

- Unabhängige Eisen (*Master*-Eisen): die Eisenkoordinaten werden so gesetzt, dass das L0-Segment (=Hauptsegment) die reale Lage einnimmt.
- Abhängige Eisen (*Slave*-Eisen): die *Slave*-Eisen werden entlang der *Master*-Eisen aufgefädelt, typischerweise so, dass sich die Eisen im Kern schneiden.

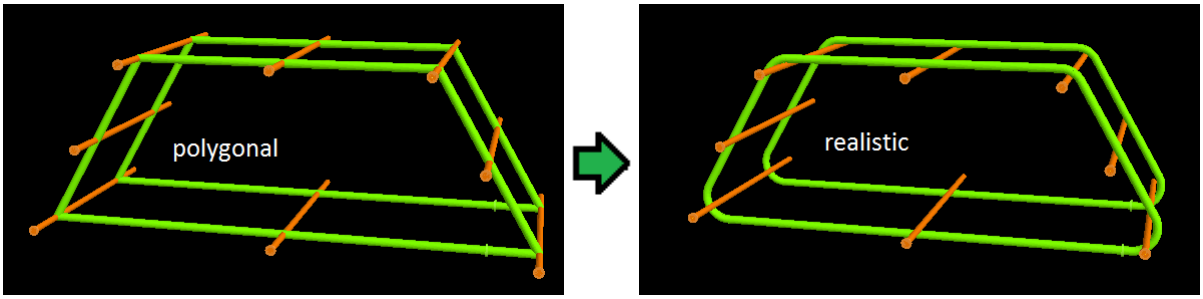
Master/*Slave*-Beziehungen können entweder explizit durch Leiteisen-Staffelmuster definiert werden[32], oder implizit über die Bewehrungsart: zwei Eisen mit Bewehrungsart 1 und 2 stehen in einer *Master*/*Slave*-Beziehung, wenn sie sich im Eisenkern schneiden (Das Bewehrungsart-1-Eisen ist hierbei der *Master* und das Bewehrungsart-2-Eisen ist der *Slave*)[33].

Wenn man die Eisen so angibt, dass sie sich im Kern schneiden, benötigt man eine formale Zusatz-Regel, die festlegt, wie die Eisen-Lagen real zueinander versetzt sind. Für *Master*/*Slave*-Eisen die sich im Kern schneiden, soll daher folgendes gelten:

Bei geraden *Master*-Eisen liegen die *Slave*-Eisen oben (also Bewehrungsart 1 unten, Bewehrungsart 2 oben). Bei gebogenen *Master*-Eisen liegen die *Slave*-Eisen *innen*[34].

3.10.1.3 ShapeMode "polygonal"

Die *polygonal*-Darstellung ist der *schematic*-Darstellung ähnlich, und die meisten oben beschriebenen Regeln können unverändert übernommen werden. Im Gegensatz zur *schematic*-Darstellung werden aber nicht die üblichen Eisenlängen angegeben, sondern die Kantenlängen des Polygons, das die reale Biegeform abdeckt.



Für Biegungen bis zu 90° stimmen die *schematic* und die *polygonal*-Darstellungen überein. Die Darstellungen unterscheiden sich aber bei spitzen Biegungen: die *polygonal*-Darstellung bleibt auch bei spitzen Biegewinkeln formgetreu zum realen Eisen, während die *schematic*-Darstellung die äußeren Proportionen verändert[35].

Für "Abrundungsbiegungen" an den Eisenenden, ist die *polygonal*-Darstellung aber oft unnatürlich und für 180°-Biegungen ist sie sogar unmöglich (die Polygonschenkel wären dann unendlich lang). In diesem Fall kann man einen spitzen Biegewinkel α in zwei Winkel α_1 und α_2 aufteilen, und zwischen diesen Biegungen eine Zusatzkante einfügen, die folgende Länge hat:

$$L_k = R \cdot \left(\tan\frac{\alpha_1}{2} + \tan\frac{\alpha_2}{2}\right), \qquad \alpha_1 + \alpha_2 = \alpha$$

Hierbei ist R der Biegeradius im Eisenkern.

Typischerweise ist α_1 durch eine äußere Formgebung des Polygons gegeben, und α_2 vervollständigt die Biegung, um auf den gewünschten Gesamtwinkel α zu kommen. Die dazwischen künstlich einzuführende Polygonkante L_k kann wie oben beschrieben ermittelt werden. Nach der Umrechnung in die *realistic* Darstellung reduziert sich die Zusatzkante auf ein Segment der Länge 0, das dann sinnvollerweise weggelassen wird – die beiden Biegewinkel α_1 und α_2 verschmelzen dann wieder zu einem Gesamtwinkel α.

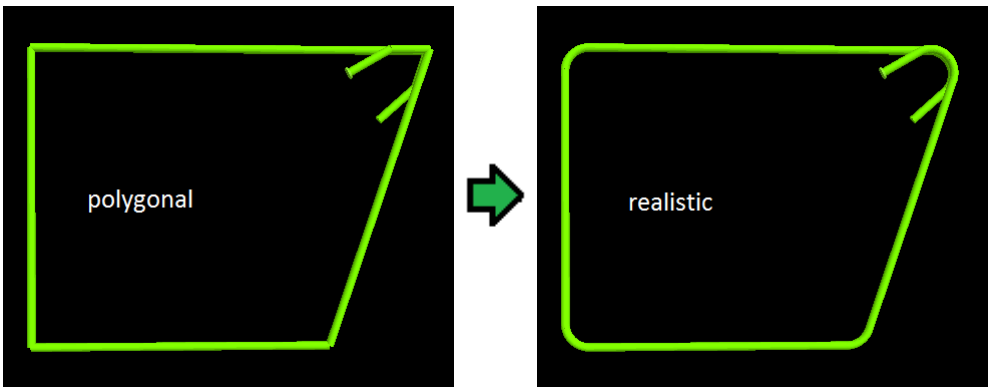
---

[32] Leiteisen-Staffelmuster: siehe Abschnitt 3.12.1.

[33] Analog dazu kann ein Bewehrungsart-4-Eisen ein *Slave* zu einem Bewehrungsart-2-Eisen sein. Diese mehrstufige Abhängigkeit wird aber nur bei sehr komplizierten Körben verwendet. (Für eine Definition der Bewehrungsarten siehe Abschnitt 3.10.2).

[34] In praktischen Fällen ist bei Biegeformen klar ersichtlich, wo *innen* und *außen* ist. Eine allgemeine mathematisch präzise Definition von *außen* und *innen* kann wie folgt festgelegt werden:
Seien $\vec{v_i}$ die Vektoren der Eisensegmente, und $\vec{a} := \sum_{i=0}^{n-2} \vec{v_i} \times \overrightarrow{v_{i+1}}$. Dann zeigt $\vec{a} \times \vec{v_i}$ nach *innen*.

[35] In obigem Beispiel hat der Korb in der *polygonal*- und in der *realistic*-Darstellung dieselbe Höhe; in der *schematic*-Darstellung hätte er eine geringere Höhe.



Unter Einbeziehung dieser Winkelaufteilung wird die *polygonal*-Darstellung zu einer sehr universellen Form, die der *schematic*-Darstellung überlegen ist[36].

Die Abhängigkeit des L_k vom R bringt eine gewisse Redundanz in die Daten, da L_k aus R in eindeutiger Weise hervorgeht. Tatsächlich kann man L_k auch kleiner wählen, oder sogar auf 0 setzen, denn das datenempfangende System weiß, dass jeder Polygonschenkel mindestens die oben angeführte Länge haben muss. Man hat dann aber, so wie in der *schematic*-Darstellung, für manche Eisensegmente eine Abweichung des Polygons vom realen Eisen. Das betrifft typischerweise aber nur Endabrundungen, so dass die Gesamtform des Eisens unverfälscht wiedergegeben wird.

3.10.1.4 Automatische Ermittlung des ShapeMode und gemischte Darstellungen

Oft wird kein *ShapeMode* angegeben, insbesondere dann, wenn die Daten aus UNICAM- oder BVBS-Dateien importiert werden. Da unterschiedliche CAD-Systeme mit unterschiedlichen Darstellungsformen arbeiten, müssen Daten ohne *ShapeMode*-Angabe als unvollständig angesehen werden[37].

In UNICAM-Dateien wird der *ShapeMode* manchmal (in Erweiterung der offiziellen Definition) in das Reservefeld RODSTOCK Zeile 5, Spalten 9-11 abgebildet:

- 000 = *undefined*
- 001 = *schematic*
- 002 = *polygonal*

Einige CAD-Systeme verwenden auch eine **gemischte Darstellung**: die gebogenen Eisen werden im *schematic*-Mode dargestellt, aber die geraden Eisen werden an ihre reale Raumposition gesetzt. Die Eisen schneiden sich also nicht im Eisenkern und werden in ihrer Position daher nicht über eine *Master/Slave*-Beziehung korrigiert.

3.10.2 ReinforcementType (Bewehrungslagen)

3.10.2.1 Definition der Bewehrungsarten

Die Bewehrungstypen folgen weitgehend der UNICAM-Definition; es wurden jedoch Zusatzdefinitionen für die Korbfertigung eingeführt:

0 = ohne Definition.

---

[36] Die genannte Winkelaufteilung kann in der *schematic*-Darstellung ebenso durchgeführt werden; man kann dadurch vermeiden, dass Biegewinkel über 90° angegeben werden müssen, und hätte dann vollkommene Übereinstimmung zwischen der *schematic*- und der *polygonal*-Darstellung. Die Überlegenheit der *polygonal*-Darstellung entsteht aber durch die Möglichkeit spitze Biegewinkel durch spitze Polygon-Winkel formgetreu anzugeben – das ist in der *schematic*-Darstellung nicht möglich.

[37] Es gibt sinnvolle Ansätze, den beabsichtigten *ShapeMode* systematisch zu erraten: Daten mit expliziter Angabe von Biegeradien oder Biegerollendurchmessern verwenden typischerweise die *realistic* Form; Daten ohne diese Angaben sind meistens in *schematic* Form. Diese Regel hat aber keine uneingeschränkte Gültigkeit.

1 = erste Eisenlage; **bei Korbfertigung: Bügel**.

2 = zweite Eisenlage; **bei Korbfertigung: Längseisen**.

3 = Spieße.

4 = übrige Bewehrung (dritte Eisenlage).

5 = obere Bewehrung erste Eisenlage.

6 = obere Bewehrung zweite Eisenlage.

7 = obere Bewehrung übrige Eisen (obere Bewehrung dritte Eisenlage).

8 = lose Eisen (nicht verschweißt, nicht an andere Eisen gebunden).

Für Extirons entspricht der ReinforcementType dem UNICAM Extiron-Type.

3.10.2.2 Festlegung der Bewehrungslagen

Die genaue Bedeutung der einzelnen Bewehrungsarten hängt vom konkreten Herstellungsverfahren ab. Falls es in einer Anlage mehrere Herstellungsverfahren gibt, hängt die Bedeutung des *ReinforcementType*-Feldes daher vom *MeshType* des *Steel*-Blocks ab (Siehe auch Abschnitt 3.9.5).

Allgemein lassen sich aber folgende Empfehlungen für die Verwendung der Bewehrungsarten anführen:

Bewehrungsarten 1, 2 und 4:

Für flache Matten wählt man die Bewehrungsart direkt über die Solllage der Eisen: jene Eisen, die in der Produktionspalette unten liegen (= außen in der Doppelwand), bilden die Lage 1; die darüberliegenden Eisen bilden die Lage 2.

Für gebogene Matten und Körbe wird folgende Festlegung empfohlen:

a) Falls nur in einer Richtung gebogen wird, so sollte es die 1. Lage sein, die gebogen wird.

b) Falls in beide Richtungen gebogen wird, soll es möglich sein, bei der ausgerollten Matte die 1. Lage als letzte zu biegen.

Der Punkt b) ist so zu verstehen, dass beim Ausrollen der Matte zunächst die erste Lage auszurollen ist, dann die zweite. Beim Ausrollen der ersten Lage werden die anliegenden Lage-2-Eisen mitgenommen; beim darauffolgenden Ausrollen der zweiten Lage werden *keine* Lage-1-Eisen mehr mitgenommen[38].

Die Lage 1 kann auch als *Hauptlage* oder *stabile Lage* gesehen werden: die Koordinatenkorrekturen für Drahtdurchmesser und Biegeradien werden von der Lage 1 dominiert; die Lage 2 passt sich an die Lage 1 an, nicht aber umgekehrt.

Bewehrungsart 8:

Die *Bewehrungsart 8* kennzeichnet ein Eisen als „Außenseiter“ innerhalb des betreffenden Herstellungsverfahrens. Wenn beispielsweise ein *Steel*-Block über eine Mattenschweißanlage hergestellt wird (was eventuell durch den *MesgType* festgelegt werden kann), so würde die *Bewehrungart 8* festlegen, dass das betreffende Eisen nicht mitverschweißt wird. Wenn dagegen der gesamte *Steel*-Block von vorneherein einem manuellen Herstellungsverfahren zugeführt werden soll, so ist das über den *MeshType* festzulegen.

[38] Sollte es Eisen geben, die beim Ausrollen der 2. Lage mitgenommen werden müssen, so sind diese einer 3. Lage zuzuordnen (wähle hierfür ReinforcementType = 4, "Übrige Bewehrung"). Um solche Eisen als Matteneisen produzieren zu können, benötigt man in Längs- und Querrichtung eine Balkenbiegemaschine; andernfalls sind diese Eisen als lose Eisen zu produzieren und müssen manuell verschweißt werden. Eisen, die ungebunden sind, also nicht mit anderen Eisen auszurollen oder zu biegen sind, werden dem Bewehrungstyp 8 zugeordnet.

3.10.2.3 Obere Bewehrungslagen

Die Bewehrungstypen 1, 2 und 4 bilden einen zusammenhängenden Satz von "Lagen" wie er oben beschrieben wurde. Die Bewehrungstypen 5, 6 und 7 bilden einen zweiten unabhängigen Satz, für den wieder die gleichen Regeln gelten, wie für den ersten Lagen-Satz.

Der zweite ("obere") Lagen-Satz wird nur dann zwingend benötigt, wenn man innerhalb eines *Steel*-Blocks 2 unabhängige Lagen-Sätze braucht (Lagen in verschiedenen *Steel*-Blöcken sind ohnedies unabhängig voneinander). In der Praxis dürfte das sehr selten vorkommen, da komplexe Bewehrungskörbe üblicherweise in mehrere *Steel*-Blöcke getrennt werden, die üblicherweise sogar zu unterschiedlichen Zeitpunkten gefertigt werden (Siehe z.B. "Deckelmatte" bei Massivwänden).

Die oberen Lagen werden somit hauptsächlich für die Kompatibilität zu UNICAM benötigt. PXML-Implementierungen sollten die oberen und die unteren Lagen gleichbehandeln, denn es ist durchaus möglich, einen *Steel*-Block vorzufinden, der nur obere Lagen enthält[39].

3.10.3 SteelQuality

Stahlgüte.

3.10.4 PieceCount, Diameter, X, Y, Z

Stückzahl, Durchmesser und Offsets.

3.10.5 RotZ

Für die Verlegung eines Eisens wird das Koordinatensystem (nachdem es um X/Y/Z verschoben wurde) um den Winkel ***RotZ*** um die z-Achse gedreht. Alle weiteren Richtungsangaben beziehen sich auf das so gedrehte Koordinatensystem (Siehe auch Segmentorientierung, Abschnitt 3.10.16.1).

Der zulässige Bereich für *RotZ* ist

$$RotZ \in]-180°, 180°].$$

RotZ legt die Verlege-Richtung des Eisens fest und entspricht in UNICAM in etwa dem Winkel zur x-Achse (letztere liegt jedoch im Bereich [-360, 360°[). Wenn also das erste Teilstück des Eisens (oder das gesamte Eisen) in der xy-Ebene liegt, ist *RotZ* der Winkel, den das Eisen mit der x-Achse einschließt.

Für den allgemeinen Fall ist die Festlegung von *RotZ* etwas komplizierter:

Angenommen, die Biegeebene der ersten Biegung schneide die xy-Ebene; RotZ ist dann der Winkel, den die Schnittlinie mit der x-Achse einschließt.

Genauer heißt das: wenn $\vec{v}_0$ und $\vec{v}_1$ die ersten beiden Teilstücke des Eisens sind, dann ist

$$\boxed{RotZ = arctan\left(\frac{v_{0y}v_{1z}-v_{0z}v_{1y}}{v_{0x}v_{1z}-v_{0z}v_{1x}}\right) + k\pi.}$$

(Es kann durchaus vorkommen, dass der Nenner 0 ist; das Ergebnis ist dann natürlich $\pm\frac{\pi}{2}$. Wirklich undefiniert ist das Ergebnis nur wenn Nenner und Zähler 0 sind; dieser Fall tritt aber dann ein, wenn $\vec{v}_0$ und $\vec{v}_1$ parallel oder antiparallel sind, oder beide in der XY-Ebene liegen, oder nur ein Segment existiert).

Obige Gleichung enthält noch den unbestimmten Summanden $k\pi$, wobei k gleich 0 oder ±1 ist. Tatsächlich bestimmt die angeführte Überlegung dieses Detail nicht, denn eine Richtung ist zunächst

[39] Die obere Matte ("Deckelmatte") von Massivwänden könnte beispielsweise ausschließlich obere Lagen enthalten. Für eine PXML-Implementierung soll dies jedoch irrelevant sein.

gleich gut, wie die um 180° entgegengesetzte. Um k festzulegen kann man fordern, dass die Projektion von $\vec{v}_0$ die Richtung festlegt, dass also[40]:

$$\boxed{\vec{v}_r \cdot \vec{v}_0 \geq 0 \quad \text{bei } \vec{v}_r := (cos(RotZ), sin(RotZ), 0)}$$

Herleitung der Gleichung für *RotZ*:

Sei λ derart, dass

$$\vec{v}_0 - \lambda\, \vec{v}_1 = \begin{pmatrix} x_p \\ y_p \\ 0 \end{pmatrix}.$$

Der Punkt (x_p, y_p) ist die Projektion von $\vec{v}_0$ in der XY-Ebene, wenn entlang des Vektors $\vec{v}_1$ projiziert wird. Aus der Z-Komponente der obigen Gleichung ergibt sich $\lambda = v_{0z}/v_{1z}$.

Wegen$RotZ = arctan(y_p/x_p)$ ergibt sich dann obige Beziehung für *RotZ*.

Ein Problem hat man, wie gesagt, bei $x_p = y_p = 0$. In diesem Fall ist es sinnvoll einfach nur die Projektion des ersten Teilstückes zu betrachten:

$$RotZ = arctan2(v_{0y}, v_{0x}), \text{ (bzw. } RotZ = 0 \text{ bei } v_{0x} = v_{0y} = 0).$$

Anmerkung: Grundsätzlich ist die Angabe von *RotZ* überflüssig, denn bei beliebigem *RotZ* kann man über geeignete Segmentwinkel *RotX* und *BendY* (Siehe Abschnitt 3.10.16.1) jeden beliebigen Richtungswechsel durchführen. *RotZ* wurde nur deshalb eingeführt, um den wichtigen Spezialfall der konstanten Biegeebene einfach und anschaulich behandeln zu können. Aus diesem Grund ist obige Festlegung von *RotZ* nur als Empfehlung zu verstehen. Im Prinzip steht es jedem frei *RotZ* nach Belieben zu setzen, oder auch überhaupt nicht zu verwenden (d.h. *RotZ=0* zu lassen).

3.10.6 ArticleNo

Artikelbezeichnung des Rundstahl-Eisens.

3.10.7 NoAutoProd

Gesetzt, wenn das Eisen *nicht* automatisch von der Stahlmaschine produziert werden soll.

Wird typischerweise für Lagerware genutzt, die in Standardlängen vorproduziert wurde, und folglich von der Just-in Time-Produktion ausgenommen wird.

3.10.8 ExtIronWeight

Gewicht eines *ExtIron*-Eisens in *Kg*. Wird nur für *ExtIron*-Eisen verwendet (denn diese haben i. all. keine Dimensionsangaben).

3.10.9 Bin

Fachzuweisung (z.B. für die Zuweisung eines Wagenfachs der Stahlmaschine).

3.10.10 Pos

Textfeld für Positionsangabe des Einzelstabes.

3.10.11 Note

Text-Feld für Notiz zum Eisen.

3.10.12 Machine

Textfeld für Angabe der Produktions-Maschine.

[40] Diese Einschränkung führt nicht nur zu einem intuitiv sinnvollen *RotZ*, sondern vereinfacht auch andere Berechnungen, wie man im Abschnitt 3.10.16.12 sehen wird.

Eine maschineninterne Produktionsliste kann optional auch als eigene Maschine behandelt werden. In diesem Fall wird empfohlen, folgendes Format zu wählen:

MSR:2

In diesem Beispiel ist "MSR" die eigentliche Maschinenangabe, und "2" die Nummer der Produktionsliste.

3.10.13 BendingDevice

Biegevorrichtung. Ist ein Textfeld, in dem normalerweise der Durchmesser (in mm) der zu verwendenden Biegematrize angegeben werden kann. Da es ein Textfeld ist, kann die Angabe jedoch auch allgemeiner sein (z.B. Matrizenbezeichnung). Das Feld wird in BVBS auf das Header-Feld 's' abgebildet[41].

Die Angabe der *BendingDevice* impliziert oft einen Mindest-Biegeradius. Bei einer Biegematrize mit Durchmesser *D* hat ein Draht mit Durchmesser *d* einen Mindest-Biegeradius von

$$r_{min} = \frac{D}{2} + \frac{d}{2} + k_r.$$

(k_r ist hierbei ein Radius-Korrektur-Wert für die Rückfederung, also die Differenz zwischen Radius am Draht und an der Biegematrize. Vereinfachend wird dieser Wert häufig zu 0 angenommen)[42].

Daraus ergibt sich natürlich ein potentieller Konflikt mit der Biegeradius-Angabe des Segmentes (siehe Abschnitt 3.10.16.3). In diesem Fall ist immer der größere Biegeradius relevant.

3.10.14 Spacer

Ein Spacer-Eintrag beschreibt einen einzelnen Abstandhalter. Im Gegensatz zu einigen anderen Datenformaten werden in PXML die Abstandhalter stets einzeln gelistet (es gibt nicht die Möglichkeit eine feste Teilung einzugeben).

3.10.14.1 Type

Abstandhaltertyp; entspricht typischerweise der Betondeckung in mm, dividiert durch 5.

3.10.14.2 Position

Position des Abstandhalters entlang des Eisens. Die Positionsangaben beziehen sich auf die *theoretischen* Längen.

3.10.15 WeldingPoint

3.10.15.1 WeldingOutput

Schweißleistung in %.

---

[41] In UNICAM wird das Feld (in Erweiterung der offiziellen Definition) auf das Reservefeld RODSTOCK Zeile 5, Spalten 5-7 abgebildet, wobei hier aber nur numerische Werte eingetragen werden, und auch das auf 3 Zeichen limitiert.

[42] Die Berücksichtigung von k_r ist in nur dann nötig, wenn große Genauigkeitsansprüche vorliegen. In der Praxis kann das vorkommen, wenn gebogene Eisen oder Körbe maschinell weiterverarbeitet werden. Dann muss zum einen die Gesamtgeometrie des Eisens gut bekannt sein, und diese hängt bei Biegungen über 90° eben auch von k_r ab. Vor allem müssen aber die realen Segmentlängen am ausgerollten Eisen mit hoher Genauigkeit bekannt sein. Es wird daher empfohlen, die messtechnische Ermittlung von k_r über die realen Eisenlängen zu machen. Dadurch wird nicht nur die Rückfederung berücksichtigt, sondern auch die Tatsache, dass die längenneutrale Fase der Biegung eventuell nicht exakt in der Eisenmitte verläuft (wenn sie weiter innen verläuft, wird die Eisenmittellinie beim Biegen gelängt, was über eine Erhöhung von k_r rechnerisch berücksichtigt werden kann).
Man beachte, dass k_r keine Konstante ist, sondern abhängig vom Draht. für dünne Drähte ist die Rückfederung deutlich größer als für dicke Drähte.

3.10.15.2 Position

Position des Schweißpunktes entlang des theoretischen Eisenpfades.

3.10.15.3 WeldingPointType, WeldingPrgNo

Der **WeldingPointType** enthält eine Typ-Kennung des Schweißpunktes.

Die **WeldingPrgNo** bestimmt direkt das Schweißprogramm das verwendet werden soll. Die Bedeutung dieses Wertes hängt vom *WeldingPointType* und non der verwendeten Maschine ab.

Folgende *WeldingPointType*-Werte sind im Standard definiert:

- **0:** Kein bekannter Typ
- **1:** Generischer Schweißpunkt
- **-1:** Punkt darf nicht verschweißt werden
- **-2:** Etikettier-Punkt
- **-3:** Punkt für Farbmarkierung (falls unterschiedliche Farben zur Auswahl stehen, kann das Feld *WeldingPrgNo* verwendet werden, um die Farbe zu spezifizieren).
- **-17**: Markierung für Hauptsegment („L0"-Segment). Ein Segment, das so markiert ist, soll produktionstechnisch als zentrales und stabiles Hautsegment verwendet werden.
- **-23**: **BendingStep**. Markiert einen Biegeschritt, bei dem das angegebene Segment durch eine Biegeaktion bewegt wird.

  Die *WeldingPrgNo* bestimmt, ob die Biegung am Anfang oder am Ende des Segments erfolgt: eine negative *WeldingPrgNo* gibt an, dass am Ende gebogen wird.

  Die *GroupID* kann zur Gruppierung von Biegungen und zur Identifizierung dieser Gruppen für externe Referenzen verwendet werden. Die Abfolge der *BendingSteps* innerhalb eines Stabes definiert eine angenommene Biegefolge bei der Herstellung des Stabes. Obschon die tatsächliche Biegefolge davon abweichen kann, ist die angenommene Biegefolge für die Bestimmung der Schweißpunkte zwischen den Lagen relevant.
- **-29**: **GuidedPoint**. Markiert einen Punkt auf einem „geleiteten Eisen", der mit einem „Leiteisen" so verbunden ist, dass sich das geleitete Eisen mitbewegt, wenn das Leiteisen verschoben oder aufgebogen wird.

  Die *GroupID* wird verwendet, um einem *GuidedPoint* dem jeweiligen *GuidingPoint* zuzuordnen.
- **-30**: **GuidingPoint**. Markiert einen Punkt auf einem „Leiteisen", der mit einem „geleiteten Eisen" so verbunden ist, dass sich das geleitete Eisen mitbewegt, wenn das Leiteisen verschoben oder aufgebogen wird.

  Die *GroupID* wird verwendet, um einem *GuidedPoint* dem jeweiligen *GuidingPoint* zuzuordnen.
- **-100**: **GrippingPoint**. Markiert einen Punkt, an dem ein Eisen gegriffen wird, um das Eisen oder die Matte zu transportieren.

Die *GroupID* wird verwendet, um einem *GuidedPoint* dem jeweiligen *GuidingPoint* zuzuordnen.Die negativen Werte bezeichnen Typen, die keine echten Schweißpunkte beschreiben, die aber dennoch einer ähnlichen Logik folgen.

3.10.15.4 GroupID

Das **GroupID**-Feld kann verwendet warden, um Schweißpunkte zu untereinander zu verliken. Wenn beispielsweise zwei Rundeisen miteinander verschweiß werden, so gibt es an der betreffenden Schweißstelle an beiden Eisen einen *PXML WeldingPoint*. Diese beiden *PXML WeldingPoints*

gehören zusammen und sollten über eine gleichlautende *GroupID* als zusammengehörig gekennzeichnet werden.

3.10.16 Segment

Ein *Segment* ist ein Teilstück eines Rundstahl-Eisens. Ein gerades Eisen besteht aus genau einem Segment; gebogene Eisen haben für jedes Teilstück ein Segment (d.h. bei *n* Biegungen hat man *n+1* Segmente).

Das erste Teilstück beginnt an den für das Eisen angegebenen X/Y/Z-Koordinaten; jedes weitere Teilstück beginnt am Endpunkt des vorhergehenden Teilstückes.

Die Angabe des **Type**-Attributes ist optional. Das *Type*-Attribut kann folgende Werte annehmen:

- Type = „normal“: das Segment beschreibt eine normale Strecke (das ist der Default-Typ).
- Type = „spiral“: das Segment beschreibt eine Spirale. Siehe Abschnitt 3.10.16.10

3.10.16.1 Segment-Orientation (RotX, BendY)

Für jedes Teilstück werden zwei Winkel **RotX** und **BendY** angegeben, die eine Drehung des Koordinatensystems beschreiben: das Koordinatensystem wird zunächst um die x-Achse um den Winkel *RotX* gedreht; anschließend wird um die neue negative y-Achse um *BendY* gedreht[43].

$$RotX \in [-90°, 90°] \text{ (nur bedingt[44])}.$$

$$BendY \in]-180°, 180°] \text{ (nur bedingt[45])}.$$

Die x-Achse des gedrehten Koordinatensystems gibt nun die Richtung des betreffenden Segmentes an. Die Drehungen der einzelnen Teilstücke summieren sich, d.h. beim 2. Teilstück wird (das bereit zuvor gedrehte) Koordinatensystem des 1. Teilstückes weitergedreht.

RotX dreht die Biegeebene und *BendY* entspricht dem Biegewinkel (außer für das erste Segment).

3.10.16.2 Segment-Length (L)

Die **Segmentlänge L** (Einheit mm) entspricht der sogenannten **theoretischen Länge** des Teilstückes (gemessen in der Mitte des Eisens). Falls der Biegeradius *R* größer als *0* ist, weicht die Länge *L* von der realen Eisenlänge ab.

Bei der Darstellung von *Spiralen* (Siehe Abschnitt 3.10.16.10) gibt L den *Bogenradius* an.

3.10.16.3 Bending-Radius (R)

Der **Biegeradius R** beschreibt die Krümmung der Biegungen (Einheit mm).

Falls der Wert *0* ist, können die einzelnen Systeme autonom einen Biegeradius annehmen und entsprechende Längenkorrekturen vornehmen.

Die Biegeradius-Angabe des *ersten Segmentes* ist zu ignorieren und wird stets als 0 angenommen. Dasselbe gilt für das erste Segment nach einer Spirale.

Bei Spiralen wird der Wert von *R* ebenfalls ignoriert (siehe Abschnitt 3.10.16.10).

Die Schenkellänge *L* eines Segmentes ist als *theoretische Länge* zu verstehen; bei *R>0* ist die *reale Länge* von *L* verschieden.

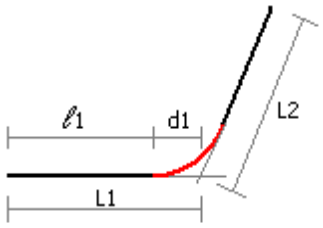
---

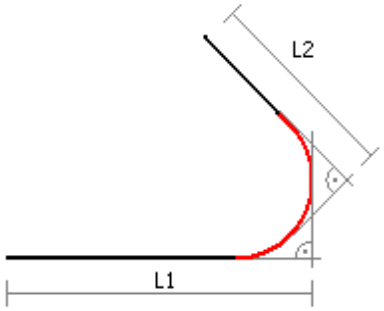
[43] Für *BendY* wird die *negative* Y-Achse verwendet, um die Biegewinkel kompatibel zu BVBS und UNICAM zu halten.

[44] Von der Definition her darf *RotX* beliebige Werte annehmen. Es ist aber empfehlenswert, Werte zwischen ±90° zu verwenden, da dieser Bereich ausreicht um jede beliebige Form darstellen zu können: *RotX*-Werte außerhalb dieses Bereiches können durch Addieren (oder Subtrahieren) von 180° in diesen Bereich hineintrasformiert werden, wenn man gleichzeitig alle folgenden *BendY* invertiert. Durch die Einschränkung auf den Bereich zwischen ±90° gewinnt die Darstellung an Eindeutigkeit und man vermeidet den Fall *RotX*=±180°, der die Darstellung in unnatürlicher Weise verkomplizieren würde, da dieser Fall auch über *RotX*=0 beschrieben werden könnte.

[45] Die Beschränkung von *BendY* auf Werte zwischen ±180° ist nur für Biegeradius *R=0* sinnvoll; bei *R>0* kann *BendY* im Bereich ±360° liegen. Um schließlich die Darstellung beliebig langer Spiralen auf einfache Weise zu ermöglichen, ist es in PXML erlaubt, für *BendY* beliebige Werte anzugeben, d.h. auch solche, die über ±360° hinausgehen.

Die Definition der theoretischen Länge ist folgenden Abbildungen zu entnehmen:





Für Winkel, die nicht größer als 90° sind, beeinflusst *R* die globale Struktur der Biegeform *nicht*, sondern führt lediglich zu einer Abrundung der Ecken. Die prinzipielle Form ist dann durch die theoretischen Längen gegeben. Bei Biegewinkel über 90° hat *R* direkten Einfluss auf die Gesamtabmessungen.

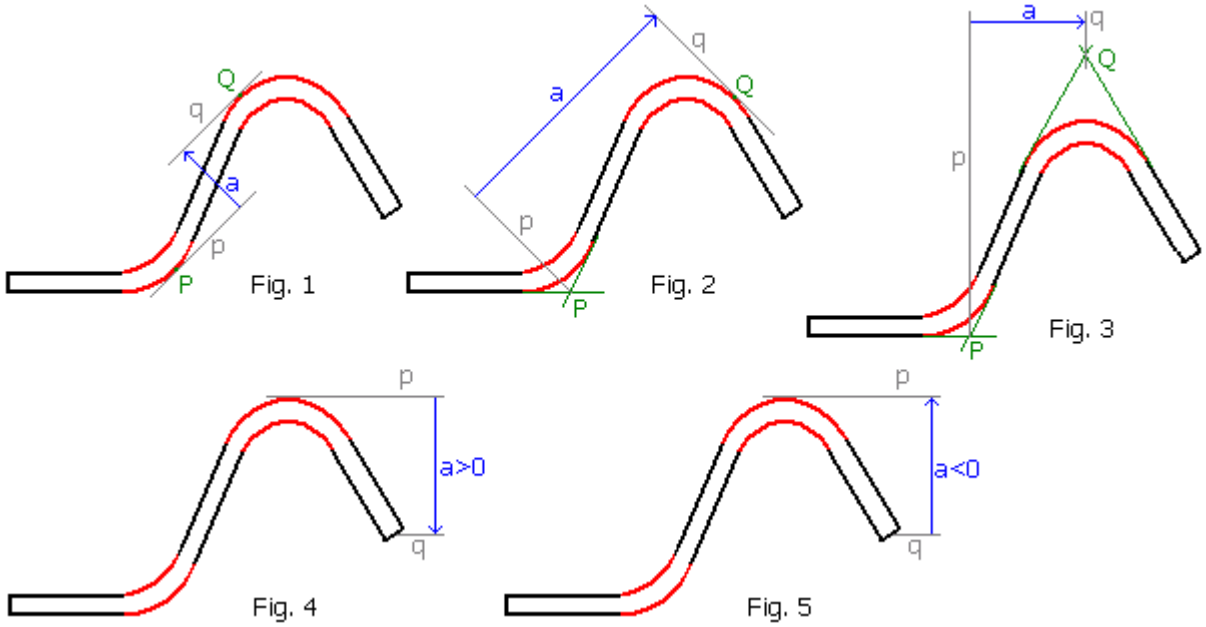
Radius-Definition über Angabe der ***BendingDevice***:

Alternativ zur direkten Radiusangabe im Segment, kann es eine implizite Biegeradius-Bestimmung über die *BendingDevice* geben (Siehe Abschnitt 3.10.13). Falls beides gegeben ist (*BendingDevice* und *R*), ist der größere Radius relevant.

3.10.16.4 Segment-Außenmaße

Normalerweise werden immer *Kernmaße* betrachtet, d.h. alle Abmessungen beziehen sich auf die Eisenmitte. In einigen Fällen (insbesondere beim Entwerfen komplexer Biegeformen) ist es sinnvoll Außenmaße zu betrachten.

Ein **Segment-Außenmaß** ist erst durch die Angabe einer **Bemaßungsrichtung** eindeutig festgelegt.



Die oben stehenden Abbildungen zeigen, wie die Segment-Außenmaße für verschiedene Bemaßungsrichtung definiert sind. Die *Bemaßungsrichtung* (Richtung des blauen Pfeiles) definiert 2 *Maßhilfslinien p* und *q* (graue Linien), die orthogonal zur Bemaßungsrichtung sind. Das *Segment-Außenmaß* entspricht dem Abstand der Maßhilfslinien *p* und *q*. Das *Segment-Außenmaß* ist *positiv*, wenn die Normale von *p* nach *q* parallel zur Bemaßungsrichtung ist; das Maß ist *negativ*, wenn die Normale von *p* nach *q* antiparallel zur Bemaßungsrichtung ist (*p* ist die Hilfslinie am *Beginn* des Segments, *q* ist die Hilfslinie am *Ende* des Segments).

Fig. 1: Falls möglich, sind *p* und *q* Tangenten (der Bogen-Außenseite).

Fig. 2: Hier kann *p* nicht Tangente sein. Stattdessen gehr *p* durch den Schnittpunkt *P* der beiden "Rand"-Tangenten des Bogens.

Fig. 3: Hier ist weder *p* noch *q* eine Tangente; die Linien werden hier über die Schnittpunkte *P* und *Q* festgelegt.

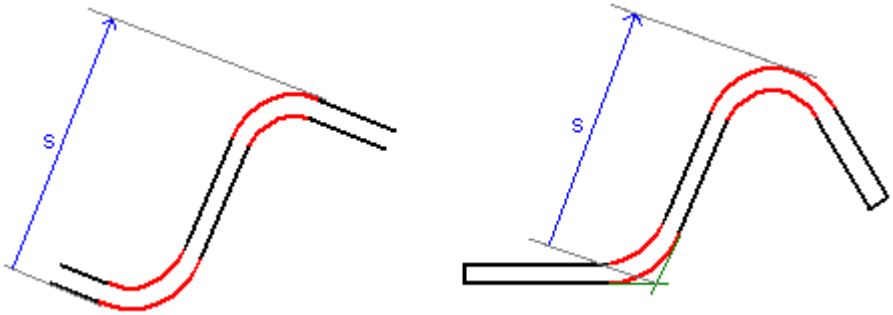
Fig. 4: Falls ein Segment an einem Ende keine Biegung hat (erstes oder letztes Segment), bezieht sich die Bemaßung auf den Eisen-Kern (Dies gilt jedoch nicht für *konventionelle Außenmaße*; siehe hierzu Abschnitt 3.10.16.8).

Fig. 5: Der positive Abstand wird stets von *p* nach *q* gezahlt (*p* an Segment-Beginn). Wenn dies entgegen der Bemaßungsrichtung ist, so ist der Abstand *negativ*.

Für Biegewinkel, die betragsmäßig größer oder gleich 180° sind, ist obige Definition mehrdeutig. In diesem Fall wird die Maßhilfslinie so gelegt, dass sie Tangente des ersten Teils der Biegung ist (so als ob der Biegewinkel unter 180° wäre).

Formale Definition: Die **Messpunkte** *P* und *Q* sind entweder Tangenten-Punkte, oder Schnittpunkte der Rand-Tangenten (wobei immer nur Bögen betrachtet werden, die kleiner als 180° sind). Das **Segment-Außenmaß** ist das innere Produkt des Vektors PQ mit dem Einheitsvektor in Bemaßungsrichtung.

3.10.16.5 *Außenlänge eines Segmentes*



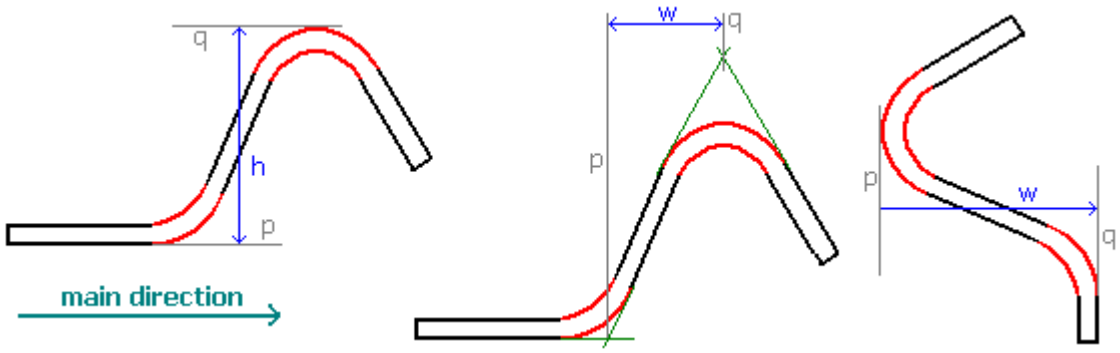
Die **Außenlänge** eines Segmentes ist jenes Segment-Außenmaß, das in Richtung des Segmentes verläuft. (Segment-Außenmaß definiert nach Abschn. 3.10.16.4.)

3.10.16.6 *Höhe und Breite eines Segmentes*

Um von **Höhe** und **Breite** eines Segmentes sprechen zu können, muss zunächst eine **Hauptrichtung** festgelegt werden. Dann gilt:

Breite = Segment-Außenmaß in Hauptrichtung.

Höhe = Segment-Außenmaß senkrecht zur Hauptrichtung (Winkel von +90° zur Hauptrichtung).



3.10.16.7 *Rechenregeln für Außenmaße*

Die Definition des Außenmaßes kann auch auf eine *Folge* von Segmenten ausgedehnt werden. Es lässt sich leicht zeigen, dass die Außenmaße **additiv** sind, d.h. das Außenmaß der Segment-Folge ist gleich der Summe der Außenmaße der Einzelnen Segmente[46].

Die Additivität kann auch verwendet werden, um ein einzelnes Segment in elementare Teilsegmente zu zerlegen; ein normales Segment kann hierbei als Folge von 3 Segmenten behandelt werden:

1) Segment mit Biegung am Beginn, gerades Teilstück der Länge 0, keine Biegung am Ende.
2) Gerades Teilstück.
3) Segment mit keiner Biegung am Anfang, gerades Teilstück der Länge 0, Biegung am Ende.

3.10.16.8 *Konventionelle Außenmaße*

Falls ein Segment-Ende keine Biegung hat, bezieht sich das Außenmaß auf den Eisen-Mittelpunkt (Eisen-Kern). Dies wurde so definiert, um möglichst einfache und klare theoretische Verhältnisse zu schaffen.

---

[46] Die Additivität der Außenmaße gilt natürlich nur dann, wenn für alle dieselbe Bemaßungsrichtung verwendet wird, und allen Außenmaßen das korrekte Vorzeichen zuordnet wird. So hat beispielsweise ein Segment, das orthogonal zur Bemaßungsrichtung steht und am Beginn 90° und am Ende -90° hat, ein Außenmaß, das gleich dem negativen Drahtdurchmesser ist.

Für den Anwender wirkt es jedoch eher unnatürlich, wenn am Beginn und am Ende des Eisens auf den Eisen-Kern Bezug genommen wird. Aus diesem Grund wird der Begriff des **konventionellen Außenmaßes** eingeführt: hierbei wird am Beginn und am Ende des Eisens nicht auf den Eisen-Kern Bezug genommen, sondern auf eine Außenseite des Eisens, und zwar auf jene Seite, die auch für die Bemaßung des anderen Segment-Endes verwendet wird[47].

3.10.16.9 Allgemeine Berechnungen zum Biege-Radius

Für die Länge b des Bogens einer Biegung um α gilt:

$$b = R \cdot |\alpha|.$$

Für das gerade Stück ℓ gilt:

$$\ell = L - d, \quad d = R\,tan(\frac{\alpha_M}{2}), \quad \alpha_M := min(\,|\alpha|,\ 90°)$$

Für die reale Länge L_R des Eisens gilt demnach

$$L_R = \ell + \frac{b}{2} = L + \frac{b}{2} - d = L + R(\frac{|\alpha|}{2} - tan(\frac{\alpha_M}{2})).$$

Das alles gilt natürlich nur, wenn $L >= d$ ist, oder anders ausgedrückt, wenn

$$L >= R\,tan(\frac{\alpha_M}{2}).$$

Es gibt verschiedene Möglichkeiten, diese Bedingung zu erzwingen; die einfachste ist wohl die, L bei Bedarf auf obigen Mindest-Wert anzuheben.

3.10.16.10 Bögen und Spiralen in klassischer Spiral-Form

Traditionell werden Bögen und Spiralen in gesonderter Form behandelt, um möglichst direkt und explizit als Bögen behandelt werden zu können. Um die Kompatibilität zu älteren Systemen zu wahren, wird auch in PXML die Möglichkeit einer gesonderten Bogendarstellung geboten, und zwar durch spezielle *spiral*-Segmente[48].

Eine *spiral*-Segment wird durch die Typ-Kennung „spiral“ gekennzeichnet.

Die Segment-Parameter haben dann folgende Bedeutung:

- ***BendY***: Windungs-Winkel der Spirale
  (kann beliebige Werte annehmen, ist also *nicht* auf ±360° beschränkt).
- ***L***: Radius der Spirale.
- ***R***: Hat keine Bedeutung und wird ignoriert[49].
- ***RotX***: Definiert die Ganghöhe der Wendel[50].
  Die Höhen-Zunahme pro Windung ist $G = L \cdot \sin\,(RotX)$.

Verbindung des Bogens zum geraden Segment:

Falls Bögen (Spiralen) mit geraden Segmenten verbunden werden, ist der Übergang Bogen-Strecke stets *ohne Biegung*. Am Beginn des Bogens muss das schon aufgrund des Datenformats so sein: da *BendY* den Bogenwinkel beschreibt, hat man keine Möglichkeit mehr, zusätzlich einen Biegewinkel anzugeben. Am Ende des Bogens, also am Beginn des darauffolgenden geraden Segmentes, wäre es

---

[47] Für ein gerades Eisen ist dies nicht definiert. Dort macht es aber keinen Unterschied ob man die Bemaßung auf den Eisen-Kern oder auf eine Außenseite bezieht.

[48] Obschon *spiral* Segmente in PXML vorgesehen sind, sollten neuere Systeme erwägen, die Bögen in normaler PXML-Darstellung zu beschreiben, wie das im Abschnitt 3.10.16.11 beschrieben wird.

[49] Beachte: die Spezifikation legt fest, dass *R* ignoriert wird, und folglich beliebige Werte annehmen kann. Es wird den Implementierungen *nicht* freigestellt, *R* anderweitig zu verwenden und dadurch die Freiheit beim Setzen von *R* einzuschränken. Diese Festlegung ist notwendig, weil der Radius *R* oft über übergeordnete Parameter (Matrizen-Parameter) für alle Segmente gesetzt wird – es soll dann nicht nötig sein, hiervon Spiralen auszunehmen.

[50] Auf die Möglichkeit, bei Spiralen ein *echtes RotX* anzugeben, wird hier bewusst verzichten, da dies in der Praxis kaum nötig sein wird. Sollte das dennoch einmal benötigt werden, muss man der Spirale ein Hilfs-Segment der Länge 0 voranstellen.

im Prinzip möglich, eine Biegung anzugeben. Dieser Freiheitsgrad soll jedoch nicht genutzt werden, d.h. das nachfolgende Segment soll *BendY=0* haben. Diese Einschränkung ist sinnvoll um am Beginn und am Ende des Bogens symmetrische Verhältnisse zu haben, und insgesamt die Rechnerischen Verhältnisse zu vereinfachen. Zudem passt diese Einschränkung auch gut zur BVBS-Kodierung und zur Funktionsweise der Biegemaschinen: in beiden Fällen müssen Biegungen am Beginn und am Ende eines Bogens über separate Segmente kodiert werden.

Spiralen und Darstellungsart:

Die Behandlung von Spiralen ist grundsätzlich nur für die *realistic*-Darstellung definiert (Siehe *ShapeMode*, Abschnitt3.10.1). Die *schematic*- und *polygonal*-Darstellungs-Konzepte lassen sich nicht sinnvoll mit dem Spiral-Darstellungs-Konzept verbinden, und die beiden Konzepte gehören auch applikationstechnisch ganz unterschiedlichen Domänen an.

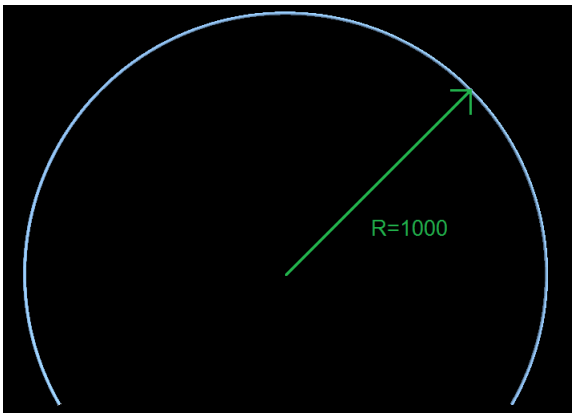
3.10.16.11 Bögen normaler PXML-Form

Ein Bogen kann in PXML auch einfach als Biegung dargestellt werden. Der Bogen ist dann nicht mehr als solcher gesondert gekennzeichnet und unterscheidet sich von einer normalen Biegung nur dadurch, dass er einen großen Biegeradius hat, und daher typischerweise nicht durch Biegen über eine Matrize hergestellt wird, sondern über andere Maschinenvorrichtungen. Tatsächlich kann es aber von Vorteil sein, die Art der maschinellen Herstellung nicht in den Daten vorwegzunehmen, sondern der Maschinensoftware zu überlassen; die PXML-Daten beschränken sich dann auch die geometrische Beschreibung des Produktes, ohne das Herstellungsverfahren vorzuschreiben.

Wenn Bögen mit normalen PXML-Segmenten beschrieben werden, müssen folgende Punkte berücksichtigt werden:

a) Am Beginn des ersten Segmentes kann es keine Biegung geben. Man braucht für die Darstellung eines Bogen-Eisens somit mindestens 2 Segmente. (Das ist der einzige wirkliche Nachteil der normalen PXML-Segmente gegenüber den *spiral*-Segmenten).
b) Wenn es vor dem Bogen eine Biegung gibt, muss diese in einem eigenen Segment untergebracht werden. (Das ist auch bei *spiral*-Segmenten so).
c) Ein Bogen ist durch seinen Radius und den Winkel bereits vollständig definiert. Der L-Wert des Segmentes ist redundant und müsste exakt den Wert $L = R\,tan(\,\alpha_M/2)$ einnehmen (Siehe Abschnitt 3.10.16.9). Um diese Redundanz in den Daten zu vermeiden, wird empfohlen, *L=0* zu setzen. (Außer dann, wenn auf den Bogen ein gerades Stück folgt, dann ist *L* entsprechend größer als $R\,tan(\,\alpha_M/2)$).
d) Die Darstellung einer Wendel-Gang-Höhe von Spiralen ist (derzeit) nicht vorgesehen.

Beispiele 1: Einfacher Bogen



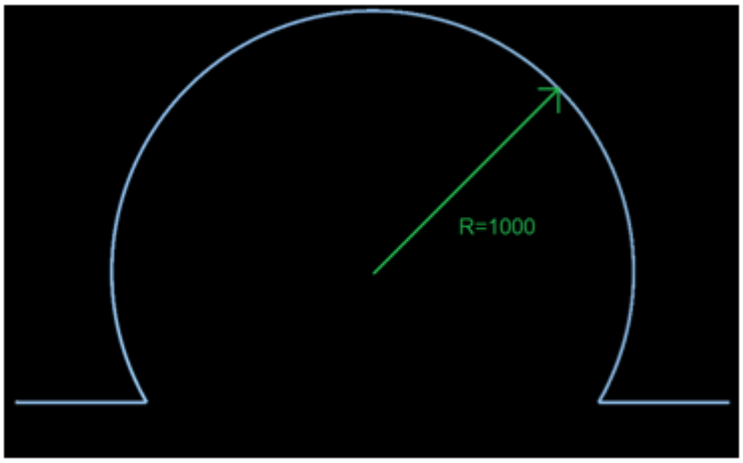
Ein Bogen mit Bogenradius 1000mm (Kernmaß) und einem Winkel von 240° wird über 2 Segmente dargestellt:

Segment 1: L=0, BendY=0, R=0

Segment 2: L=0, BendY=240, R=1000

Die Länge des Bogens ist 4189mm; diese Länge wird nicht explizit angegeben, da sie aus Radius und Winkel errechnet werden kann. Die Segmentlänge *L* wird (wie oben erwähnt) ebenfalls nicht angegeben, oder einfach auf 0 gesetzt.

Beispiele 2: Bogen mit Endaufbiegungen



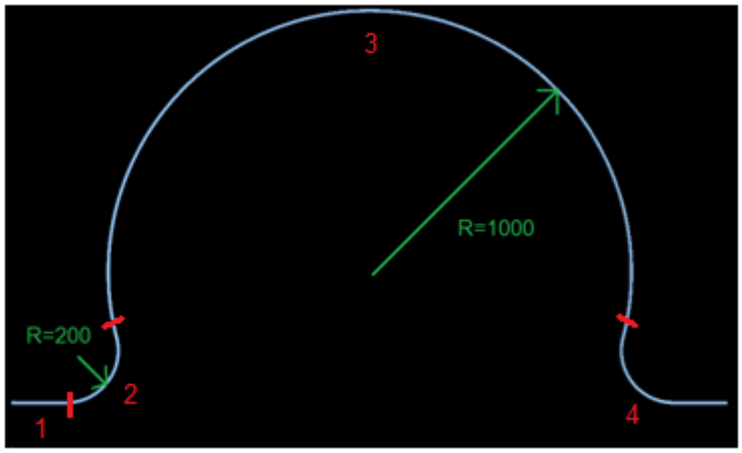
Segment 1: L=500, BendY=0, R=0

Segment 2: L=0, BendY=120, R=0

Segment 3: L=0, BendY=-240, R=1000

Segment 4: L=500, BendY=120, R=0

Beispiele 3: Bogen mit Endaufbiegungen mit Biegeradius



Segment 1: L=404, BendY=0, R=0

Segment 2: L=0, BendY=104, R=200

Segment 3: L=0, BendY=-208, R=1000

Segment 4: L=404, BendY=104, R=200

Die roten Linien und roten Ziffern im Bild beschreiben die Segmente.

Die Verkürzung der Winkel um ca. 16° (=120°-104°) kann man über folgenden Zusammenhang ermitteln:

$$\Delta\alpha = \alpha - \operatorname{acos}\frac{R_2 + R_3 \cdot \cos\alpha}{R_2 + R_3} = 120° - \operatorname{acos}\frac{200 + 1000 \cdot \cos 120°}{200 + 1000} = 15{,}52°$$

3.10.16.12 Allgemeine Berechnungen zur Koordinaten-Drehung

Es gibt insgesamt 3 Arten von Koordinaten-Drehungen: *RotZ*, *RotX* und *BendY*. Die Drehung um die z-Achse (*RotZ*) erfolgt für jeden Stab nur einmal, und ist – wie weiter oben erläutert – eigentlich frei wählbar. Die Drehungen um die x- und y-Achsen (*RotX* und *BendY*) können für jedes Teilstück angegeben werden.

Sei $\underline{v}$ die Koordinatendarstellung eines Vektors im ursprünglichen Koordinatensystem und $\underline{v}'$ die entsprechende Darstellung im gedrehten System; dann ist die Transformation durch eine **Dreh-Matrix** *D* beschrieben:

$$\underline{v}' = D \cdot \underline{v}.$$

Für die oben genannte Drehungen gilt[51]:

$$D_{RotZ} = \begin{pmatrix} \cos RotZ & \sin RotZ & 0 \\ -\sin RotZ & \cos RotZ & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D_{RotX} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos RotX & \sin RotX \\ 0 & -\sin RotX & \cos RotX \end{pmatrix}$$

$$D_{BendY} = \begin{pmatrix} \cos BendY & 0 & \sin BendY \\ 0 & 1 & 0 \\ -\sin BendY & 0 & \cos BendY \end{pmatrix}$$

$$\begin{aligned} D_{RotX,BendY} &= D_{BendY} \cdot D_{Rotx} \\ &= \begin{pmatrix} \cos BendY & -\sin RotX \cdot \sin BendY & \cos RotX \cdot \sin BendY \\ 0 & \cos RotX & \sin RotX \\ -\sin BendY & -\cos BendY \cdot \sin RotX & \cos BendY \cdot \cos RotX \end{pmatrix} \end{aligned}$$

(In diesen Angaben wurde bereits berücksichtigt, dass *BendY* eine Drehung um die *negative* y-Achse beschreibt).

Sei nun $\underline{v} = (v_x, v_y, v_z)$ die Darstellung eines Einheitsvektors, die bei der Koordinatendrehung $D_{RotX,BendY}$ in die Darstellung$\underline{x}^0 = (1,0,0)$ übergeht. Es ist also $D_{RotX,BendY} \cdot \underline{v} = \underline{x}^0$, und folglich $\underline{v} = D^T_{RotX,BendY} \cdot \underline{x}^0$. Daraus ergibt sich unmittelbar:

$$v_x = cos(BendY),\ \ v_y = -sin(RotX)\ \ sin(BendY),\ \ v_z = cos(RotX)\ \ sin(BendY).$$

Durch Kombinieren dieser Gleichungen erhält man

$$tan(RotX) = -\frac{v_y}{v_z}.$$

Wenn man nun *RotX* auf den Bereich [-90°,90°] einschränkt (was, wie weiter oben erwähnt, immer möglich ist), ergibt sich:

$$\boxed{RotX = -\,arctan\frac{v_y}{v_z}.}$$

Bei gegebenem *RotX* kann man nun sin(*BendY*) auf zweierlei Arten berechnen:

$$sin(BendY) = \frac{v_y}{-\,sin(RotX)},\quad sin(BendY) = \frac{v_z}{cos(RotX)}.$$

Da sowohl der Cosinus als auch der Sinus von *RotX* null sein kann (aber niemals beide), wählt man für die Berechnung die numerisch stabilere Variante, also jene betragsmäßig größerem Nenner.

Mit $v_x = cos(BendY)$ hat man auch cos(*BendY*) gegeben und somit kann man *BendY* eindeutig ermitteln:

$$BendY = arctan2(sin(BendY),\ cos(BendY))$$

Zusammenfassend ergibt das:

[51] Dreh-Matrizen sind stets *orthogonal*, d.h. es gilt $D^{-1}=D^T$.

$$BendY = \begin{cases} arctan\,2\left(\frac{v_y}{-\,sin(\,RotX)},\ v_x\right), & |sin(\,RotX)| \geq |cos(\,RotX)| \\ arctan\,2\left(\frac{v_z}{cos(\,RotX)},\ v_x\right), & |sin(\,RotX)| < |cos(\,RotX)| \end{cases}$$

Einen Sonderfall hat man für $v_y = v_z = 0$. Man überlegt sich leicht, dass das genau dann der Fall ist, wenn $BendY = 0$ oder $BendY = 180°$ist. Obiger "atctan"-Ausdruck zur Berechnung von *RotX* ist dann undefiniert, und tatsächlich ist *RotX* in diesem Fall nicht eindeutig bestimmbar, da bei $BendY = 0$ oder $BendY = 180°$zwei aufeinanderfolgende *RotX* um die selbe Achse drehen; es ist dann nur die Summe dieser *RotX* eindeutig festgelegt, nicht aber die Einzel-Werte. Wenn man diesen Fall nicht gesondert behandelt, wird man dennoch ein gültiges Ergebnis erhalten: *RotX* hat dann einen zufälligen Wert, der durch Rundungsfehler dominiert wird[52]; wenn man mit diesem "zufälligen" *RotX* konsequent weiterrechnet, ist das Ergebnis insgesamt noch korrekt (das Zufälligkeit des *RotX* wird im nächsten *RotX* kompensiert). Es ist aber günstiger, diesen Fall wie folgt gesondert zu behandeln:

$$\text{bei} \quad BendY_i = 0: \quad RotX_i + RotX_{i+1} \rightarrow RotX_i,\ \ 0 \rightarrow RotX_{i+1}$$

D.h. die Summe von $RotX_i$ und $RotX_{i+1}$wird vollständig in $RotX_i$ hineingenommen; $RotX_{i+1}$ auf 0 gesetzt[53].

Ein gewisses Problem hat man dagegen, wenn *BendY* (näherungsweise) ±180° ist, also im Fall $\underline{v} = (v_x, v_y, v_z) = (-1,0,0)$. Auch dann ist *RotX* mathematisch unbestimmt, ebenso wie das Vorzeichen von *BendY*. In der Praxis hat man aber einen von 0 verschiedenen Biegeradius, so dass *BendY* und *RotX* eindeutig bestimmt sind; die Biegeradius-Information ist aber in der vereinfachten Darstellung von geraden Teilsegmenten nicht enthalten. Hier muss man *RotX* und *BendY* über zusätzliche Informationen bestimmen, eventuell auch über ein abgewandeltes *v*, das, unter Einbeziehung der tatsächlichen Biegerichtung, nichtverschwindende y- und z-Komponenten liefert[54].

---

[52] Dem numerischen Sonderfall des NaN-Ergebnisses muss man auch einen reellen Wert zuordnet (z.B. 0).

[53] Diese Vorgangsweise stellt sicher, dass *RotX* innerhalb der Biegeform nur dann von 0 verschieden ist, wenn es eine echte Drehung der Biegeform gibt. Bei ebenen Biegeformen wird *RotX* nur benötigt, um den Lagewinkel des gesamten Eisens anzugeben, und man möchte dann, dass diese Angabe in $RotX_0$ erfolgt (und nicht in $RotX_1$).
Schließlich sei noch erwähnt, dass der Falls *BendY*=0 im Innern der Biegeform kaum auftritt (das wäre eine Aufeinanderfolge von 2 Segmenten ohne dazwischenliegende Biegung – das ist eigentlich ein entarteter Fall). Der erste *BendY*-Winkel ist aber sehr häufig 0, und zwar für Formen deren erstes Segment in der XY-Ebene liegt.
Der Vollständigkeit halber sei noch erwähnt, dass der Fall $v_y \neq 0$, $v_z = 0$ numerisch unproblematisch ist. Die "arctan"-Funktion zur Bestimmung von *RotX* liefert dann ±90°, wobei das Vorzeichen wiederum zufällig aus Rundungsfehler entsteht (das v_z kann rundungsfehlerabhängig geringfügig positiv oder geringfügig negativ sein). Diese Zufälligkeit wird dann in den folgenden Berechnungen wieder kompensiert und stellt kein Problem dar, da kein "künstlicher" von 0 verschiedener Wert eingefügt wurde.

[54] Für das erste Segment gilt dies Überlegung nicht, da hier der reale Biegeradius 0 ist (*BendY* gibt beim ersten Segment nur eine Richtung angibt, und keine Biegung). Das ist aber kein Problem, da der Fall *BendY*=±180° beim ersten Segment ohnedies vermieden werden kann, wenn man die Einschränkung aus Abschnitt 3.10.5 beachtet, die fordert, dass *RotZ* so gewählt wird, dass $\vec{v}_r \cdot \vec{v}_0 \geq 0$; hierbei ist $\vec{v}_0$das erste Segment und $\vec{v}_r := (cos(\,RotZ), sin(\,RotZ), 0)$. Denn es lässt sich leicht zeigen, dass diese Bedingung äquivalent zu $\cos(BendY_0) \geq 0$ ist, also zu $BendY_0 \in [-90°, 90°]$.

3.10.16.13 Konvertierung von und zu UNICAM

In UNICAM gibt es die Einschränkung, dass das erste Segment in der XY-Ebene liegen muss. Unter Umständen muss in UNICAM daher ein zusätzliches horizontales **Dummy-Segment** eingeführt werden, dessen Länge 0 ist. Dieses *Dummy-Segment* wird in PXML nicht benötigt[55]. Ein weiterer Unterschied zwischen UNICAM und PXML liegt in der Biege-Ebene: während in UNICAM alle Biegungen in einer Ebene liegen müssen, kann in PXML vor jeder Biegung eine Drehung der Biege-Ebene vorgenommen werden (*RotX*); bei der Konvertierung PXML → UNICAM geht dieser Freiheitsgrad natürlich verloren.

Im Folgenden werden folgende Benennungen für die UNICAM-Werte verwendet:

α_{toX}	=	Winkel zur X-Achse
α_L	=	Lagewinkel der Biege-Ebene (0 für vertikale Biege-Ebene)[56]
d_i		Länge des Segmentes i (i=0, 1, 2, ...)
α_i	=	Winkel am Ende des Segmentes i (i=0, 1, 2, ...)

UNICAM → PXML ohne *Dummy-Segment* ($d_0 \neq 0$):

$$RotZ = \alpha_{toX}$$
$$RotX_0 = -\alpha_L, \quad BendY_0 = 0, \quad L_0 = d_0$$
$$RotX_i = 0, \quad BendY_i = \alpha_{i-1}, \quad L_i = d_i, \quad i \geq 1$$

UNICAM → PXML mit *Dummy-Segment* (d0=0):

$$RotZ = \alpha_{toX}$$
$$RotX_0 = -\alpha_L, \quad BendY_0 = \alpha_0, \quad L_0 = d_1$$
$$RotX_i = 0, \quad BendY_i = \alpha_i, \quad L_i = d_{i+1}, \quad i \geq 1$$

PXML → UNICAM, allgemeine Überlegungen:

Bei Verwendung eines *Dummy-Segments* lässt sich PXML recht gradlinig zu UNICAM konvertieren. Die Sonderfälle ohne *Dummy-Segment* müssen dagegen gesondert behandelt werden, da hier die Information über $BendY_0$ nicht direkt übertragen werden kann (das *Dummy-Segment*, das diese Information tragen könnte, existiert in diesen Fällen nicht). Die Information über $BendY_0$ muss folglich irgendwie in die anderen Größen hineingenommen werden (der Fall ohne *Dummy-Segment* ist eben dadurch charakterisiert, dass das möglich ist).

Ferner wird man stets zwischen einfachen Eisen und gebogenen Eisen unterscheiden müssen: einfach Eisen haben keine Biege-Ebene, gebogene Eisen haben eine Biege-Ebene, die durch die erste echte Biegung ($BendY_1$) bestimmt wird. Natürlich muss bei der Konvertierung zu UNICAM angenommen werden, dass alle echten Biegungen in derselben Ebene liegen.

PXML → UNICAM bei nur einem Segment:

Sonderfall $RotX_0 = \pm 90°$ oder $BendY_0 = 0°$ (kein *Dummy-Segment* nötig):

$$\alpha_L = 0$$
$$\alpha_{toX} = RotZ - BendY_0\, sgn(\sin RotX_0)$$
$$d_0 = L_0, \quad \alpha_0 = 0$$

Sonderfall $BendY_0 = \pm 180°$ (kein *Dummy-Segment* nötig, aber Richtungsinversion):

$$\alpha_L = 0$$

[55] In PXML ist es zwar prinzipiell erlaubt Segmente mit Länge 0 zu haben, es ist jedoch nicht notwendig (und daher auch nicht sinnvoll) ein horizontales Dummy-Segment zu verwenden.

[56] Hier wird α_L im Bereich]-180°,180°] angenommen. In der UNICAM-Datei wird, je nach Format, entweder α_L direkt abgespeichert, oder ein entsprechender Winkel im Bereich [0, 360°[.

$$\alpha_{toX} = RotZ + 180°$$
$$d_0 = L_0, \quad \alpha_0 = 0$$

Alle anderen Fälle (*mit* Dummy-Segment):

$$\alpha_L = -RotX_0$$
$$\alpha_{toX} = RotZ$$
$$d_0 = 0, \quad d_1 = L_0, \quad \alpha_0 = BendY_0, \quad \alpha_1 = 0$$

<u>PXML → UNICAM bei mehreren Segmenten:</u>

Da die Biege-Ebene durch die erste Biegung bestimmt wird, ist es sinnvoll, *BendY1*≠0 vorauszusetzen. Falls die erste Biegung entartet ist (Biegewinkel 0), werden die anliegenden Segmente zu einem Segment zusammengefasst (datenmäßig oder theoretisch).

$$\alpha_L = -\,arcsin(\,sin\,R\,otX_0\,cos\,R\,otX_1 + cos\,B\,endY_0\,cos\,R\,otX_0\,sin\,R\,otX_1)$$

$\alpha_{toX} = \arg(a,\ b)$, mit

$$a = \cos RotX_0 \cos RotX_1 \cos RotZ$$
$$-\sin RotX_1 \cdot (\cos BendY_0 \cos RotZ \sin RotX_0 + \sin BendY_0 \sin RotZ)$$
$$b = \cos RotZ \sin BendY_0 \sin RotX_1$$
$$+\sin RotZ \cdot (\cos RotX_0 \cos RotX_1 - \cos BendY_0 \sin RotX_0 \sin RotX_1)$$

Bei $RotX_0 = \pm 90°$ oder $BendY_0 = 0$ oder $BendY_0 = \pm 180°$ (<u>kein Dummy-Segment</u>):

$d_i = L_i, \quad \alpha_i = BendY_{i+1}, \quad i \geq 0$ (der letzte Winkel ist 0)

Bei $RotX_0 \neq \pm 90°$ und $BendY_0 \neq 0$ und $BendY_0 \neq \pm 180°$ (<u>mit Dummy-Segment</u>):

$$d_0 = 0$$

$\alpha_0 = arg(u, v)$, mit

$$u = cos\,B\,endY_0\,cos\,R\,otX_0\,cos\,R\,otX_1 - sin\,R\,otX_0\,sin\,R\,otX_1$$
$$v = cos\,R\,otX_0\,sin\,B\,endY_0$$

$d_i = L_{i-1}, \quad \alpha_i = BendY_i, \quad i \geq 1$ (der letzte Winkel ist 0)

Der Fall ohne*Dummy-Segment* bedarf jedoch noch einer Vorverarbeitung. Vor dem Export muss die Biegeform so transformiert werden, dass folgende Bedingungen erfüllt sind:

$$RotX_0 \in [-90°, 90°]$$
$$BendY_0 = 0$$
$$RotX_1 = 0$$

Diese **Dummy-Rektifizierung** kann wie folgt bewerkstelligt werden:

Dummy-Rektifizierung bei $RotX_0 = \pm 90°$:

$$RotZ_{new} = RotZ_{old} - BendY_{0,old}\,sgn(\,sin\,R\,otX_{0,old})$$
$$BendY_{0,new} = 0$$
$$RotX_{0,new} = RotX_{0,old} + RotX_{1,old}$$
$$RotX_{1,new} = 0$$
$$RotX_{0,new} \rightarrow [-90°, 90°]$$

Dummy-Rektifizierung bei $BendY_0 = 0$:

$$RotX_{0,new} = RotX_{0,old} + RotX_{1,old}$$
$$RotX_{1,new} = 0$$
$$RotX_{0,new} \rightarrow [-90°, 90°]$$

Dummy-Rektifizierung bei $BendY_0 = \pm 180°$:

$$RotZ_{new} = RotZ_{old} + 180°$$
$$BendY_{0,new} = 0$$

$$RotX_{0,new} = 180° - RotX_{0,old} + RotX_{1,old}$$
$$RotX_{1,new} = 0$$
$$RotX_{0,new} \rightarrow [-90°, 90°]$$

Hierbei ist

$$RotX_{0,new} \rightarrow [-90°, 90°]$$

eine Transformation, die dafür sorgt, dass $RotX_0$ in diesem eingeschränkten Winkelbereich liegt. Falls nötig, muss man also $RotX_0$ um $\pm 180°$ verändern; dies kompensiert man, indem alle *BendY*-Werte invertiert ($BendY_0$ ist hiervon nicht betroffen, da dieser Wert zuvor bereits auf 0 gesetzt wurde).

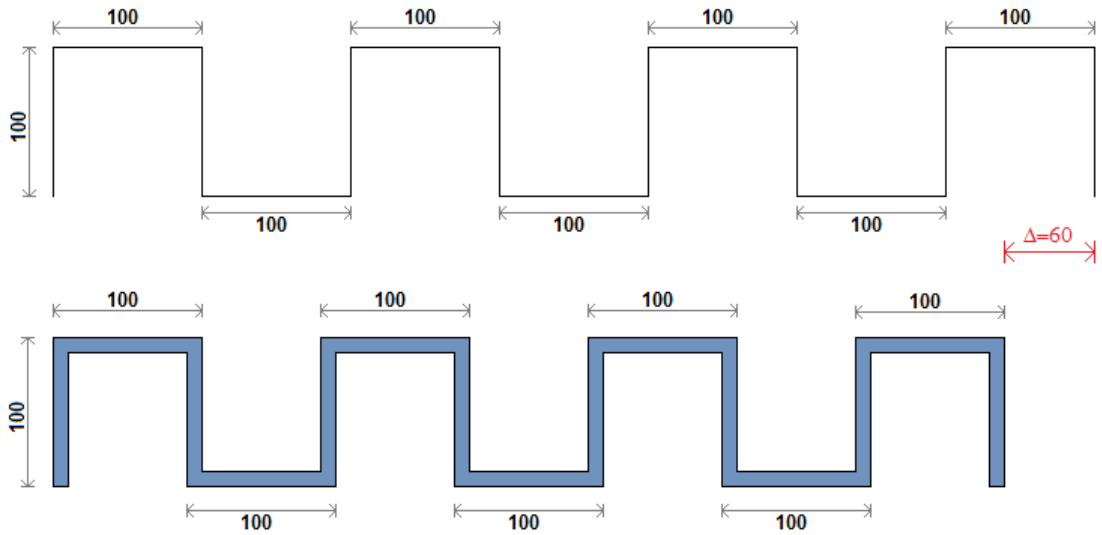
3.10.16.14 Konvertierung von und zu BVBS-BF2D

Im BVBS-Format gibt es die Möglichkeit für jede Biegung einen Biege-Radius anzugeben. Einige Maschinen interpretieren – in Abweichung von der BVBS-Spezifikation – eine Biegung mit Radius-Angabe grundsätzlich als Spirale (eine eventuelle Längen-Angabe wird dann als Bogenlänge aufgefasst). Es wird daher empfohlen, beim Export nach BVBS die Biege-Radien nicht in den Geometrie-Block zu schreiben, sondern durch die Biegerollen-Angabe im Header-Block zu spezifizieren: Der Biegerollen-Durchmesser wird auf das das Doppelte der größten Biegeradius-Angabe gesetzt. Die Radius-Angaben für Spiralen müssen natürlich im Geometrie-Block als Radius angegeben werden; der Wert des Spiral-Radius wird dem L-Parameter des PXML-Segmentes entnommen (siehe Abschnitt 3.10.16.10).

3.10.16.15 Konvertierung von und zu BVBS-BF3D

Die BVBS-Spezifikation für BF3D ist (im Gegensatz zu BF2D) alles andere als vollständig und eindeutig[57]. Es verwundert daher nicht, dass viele unterschiedliche, untereinander inkompatible, Implementierungen im Umlauf sind.

Uneinheitlich ist beispielsweise die Auffassung, ob in BF3D Außenmaße oder Eisenkernmaße zu betrachten sind. In BVBS sind ja an sich Außenmaße üblich. Mehrheitlich ist aber die Auffassung vorherrschend, dass für BF3D Kernmaße zu verwenden wären. Tatsächlich ist die Verwendung von Außenmaßen in BF3D auch eher seltsam, da die BF3D-Vektoren dann nicht mehr im oder am Eisempfad liegen, wie man an folgendem Beispiel sieht:



[57] Wir beziehen uns hier auf die allgemein verwendete BVBS-Spezifikation aus dem Jahre 2000 – eine jüngere ist uns nicht bekannt.

(Oben ist die BF3D-Form dargestellt, unten die reale Eisenform – wenn man annimmt, in BF3D würden Außenmaße angegeben).

Uneinheitlich ist ferner die Darstellung der Biegewinkel über 90°. Hier gibt es mindestens 3 verwendete Varianten:

a) Auffassung der BF3D-Vektoren als reine Längen- und Richtungsvektoren der Eisensegmente (Ist jedoch bei Biegewinkeln von 180° nicht mehr eindeutig).
b) Aufteilung des Winkels in Teilwinkel, mit jedem Teilwinkel kleiner gleich 90°.
c) Polygonale Darstellung (analog der polygonalen Darstellung in PXML). Ab einer bestimmten Winkelgröße (z.B. 135°) wird der Biegewinkel in Teile unterteilt (Denn Biegewinkel nahe an 180° sind in dieser Darstellung nicht möglich).

Eine weitere ungeklärte Frage ist, wie in BF3D die Absolut-Position des Eisens dargestellt werden soll. Sind die "Stabblock"-Einträge zu verwenden? ("X", "Y" oder "E"?). Es gibt Implementierungen, die hierfür eigene private Felder x/y/z im "Private"-Block einführen.

Aufgrund der genannten Auffassungsunterschiede ist es also nicht möglich eine einheitliche Definition für BF3D anzugeben. Wenn man aber versucht, die am häufigsten verwendeten BF3D-Auffassungen unter einer gemeinsamen Sichtweise zu vereinen, so ergibt sich folgende Festlegung, die zumindest kompatibel zu vielen wichtigen existierenden Implementierungen ist:

1) BF3D betrachtet Kernmaße (also nicht Außenmaße).
2) Die Geometrie der BF3D-Vektoren entspricht jener der polygonalen Darstellung in PXML. Biegewinkel über 90° können, so wie das für die polygonale Darstellung in PXML beschrieben wurde, aufgeteilt werden. Biegewinkel nahe 180° müssen aufgeteilt werden. Umgekehrt gilt (wie für die polygonale PXML-Darstellung), dass kurze Segmente zwischen zwei Biegungen als reine Darstellungshilfssegmente Aufzufassen sind; an der Maschine sind die beiden Biegungen wieder zu einer einzigen Biegung zusammenzuziehen.
3) Falls Absolutkoordinaten des Eisens angegeben werden sollen, erfolgt das im "Private"-Block mit den Feldern "x", "y" und "z".

3.10.16.16 Konvertierung von und zu 3D-BF2D

Aufgrund der großen Schwächen von BF3D ist es naheliegend, auf BF3D ganz zu verzichten und stattdessen ein geringfügig erweitertes BF2D zu verwenden. Tatsächlich reicht es aus, in BF2D die Möglichkeit vorzusehen, ein *RotX* anzugeben, natürlich nur in jenen Fällen, in denen das benötigt wird. Die hier vorgeschlagene Erweiterung sieht vor, beim Biegewinkel-Parameter in BF2D optional einen durch „~“ getrennten *RotX*-Wert anzugeben. Beispiel:

w30~90@

Beschreibt eine Biegung mit *RotX*=90° und *BendY*=30°.

Es ist hierbei aber für die Kompatibilität mit dem Standard-BF2D zwingend notwendig, den *RotX*-Zusatz wegzulassen, wenn *RotX* (fast) gleich 0 ist.

3.10.17 Kanonische Darstellung des Eisens

Wie bereits erwähnt, hat eine bestimmte Stabgeometrie keine eindeutige *RotZ/RotX$_i$/BendY$_i$*-Darstellung: verschiedene Kombinationen von *RotZ/RotX$_i$/BendY$_i$* können zur gleichen Geometrie führen. Wir haben aber ein paar zusätzliche Empfehlungen eingeführt, die schließlich zu einer recht eindeutigen Darstellung führen und wir wollen diese Regeln hier etwas formalisieren. Es muss betont werden, dass es keine Verpflichtung zur Einhaltung dieser Regeln gibt. Alle anderen äquivalenten Darstellungen sind ebenfalls gültig; die hier beschriebenen ausgezeichneten Formen dienen nur dazu, eine eindeutige Darstellung zu definieren, die zudem besonders leicht von Legacy-Systemen verarbeitet werden können, die oft nur 2D-Daten verstehen können.

Die Regeln sind[58]:

1) $RotX_1 = 0.$ [59]
2) $RotX_i \in [-90°, 90°]$ für $i > 0.$
3) $BendY_0 \in [-90°, 90°].$
4) Halte eine (und nur eine) der beiden folgenden Bedingungen ein:
   a. $RotX_0 \in [-90°, 90°]$ (→ **Legacy Canonical** Form)
   b. $BendY_1 > 0$ (→ **Separation Canonical** Form) [60].

Wenn ein Bewehrungseisen diese Bedingungen mit der Variante **4a** erfüllt, sprechen wir von der **Legacy Canonical** Form. Wenn die Bedingungen mit Variante **4b** erfüllt sind, sprechen wir von der **Separation Canonical** Form.

Die *Legacy Canonical* Form weist die größte Kompatibilität mit Legacy-Systemen auf, die nur 2D-Formen verarbeiten können, da in dieser Form die Verwendung der *RotX$_i$* (einschließlich *RotX$_0$*) minimiert wird. In dieser Konvention kommt man für den Großteil der in der Praxis vorkommenden Eisen ganz ohne *RotX* aus.

Die Bedingung 4a ist aber aus fundamentaler Sicht etwas problematisch, da sie nicht invariant gegenüber Änderungen der räumlichen Platzierung des Eisens ist: Eine Änderung der Rotations-Platzierung eines Eisens kann die Bedingung 4a aufheben (auch wenn sie vorher erfüllt war); um sie dann wieder zu erfüllen, kann es notwendig sein, die interne Darstellung des Stabes zu ändern (indem alle *BendY$_i$* invertiert werden).

Dieses Problem kann man vermeiden, indem man an Stelle der Einschränkung *4a* die Einschränkung *4b* einführt: man hat dann eine eindeutige Darstellung in der die „innere“ Eisenform und das „äußere“ Stab-Placement getrennt und unabhängig sind[61].

Einschränken von *BendY$_0$* oder *RotX$_0$*:

In obiger Festlegung wird *BendY$_0$* auf [-90°,90°] eingeschränkt. Ohne diese Bedingung könnte man *RotX$_0$* auf [-90°,90°] einschränken, ohne die genannte Drehinvarianz zu verlieren. Dann würden aber beispielsweise ganz normale in x-Richtung ausgerichtete Kröpfungseisen eine Darstellung mit *RotZ=BendY$_0$*=180° hervorrufen, was allgemein als störend empfunden würde. Die *Separation Canonical Form* wurde daher so definiert, dass eher *RotX$_0$*=180° akzeptiert wird als *BendY$_0$*=180°.

Wenn aber, in speziellen Legacy-Anwendungen, die Einschränkung von *RotX$_0$* wichtiger ist wird als jene von *BendY$_0$*, kann man sich folgende Tatsache zunutze machen:

---

[58] Wir setzen hier voraus, dass alle $BendY_i \neq 0$ für alle $i > 0$. Ohne diese Bedingung könnte man an jeder Stelle Zusatzsegmente einfügen, denen keine reale Geometrie entspricht. Falls es solche Segmente gibt, kann man sie immer wegbringen, indem man sie mit dem vorhergehenden Segment zusammenzieht.

[59] D.h. die erste Biegeebene dominiert die Darstellung. Da die meisten Stäbe nur eine Biegeebene haben, wird die Drehung der Biegeebene für diese Fälle vollständig vermieden.

[60] Wenn es nur ein Segment gibt, soll die Bedingung *4b* immer als erfüllt betrachtet werden.

[61] Die Bedingung *BendY$_1$>0* ist nicht für den Biegewinkel selbst wichtig. Diese Einschränkung dient dazu *RotZ*, *RotX$_0$*, *BendY$_0$* eindeutig zu machen. Ohne diese Einschränkung wäre es immer möglich, *RotX$_0$* um 180°zu verändern und dies durch Invertieren aller *BendY$_i$* zu kompensieren.

$$D_{BendY}(BendY_0) \cdot D_{RotX}(RotX_0) \cdot D_{RotZ}(RotZ) \\ = D_{BendY}(BendY_0 + 180°) \cdot D_{RotX}(180° - RotX_0) \cdot D_{RotZ}(RotZ + 180°)$$

D.h. man kann $RotX_0$ auf $RotX_0$–180° ändern, wenn man gleichzeitig $BendY_0$ und $RotZ$ um 180° erhöht. Dadurch kann man eine *Separation Canonical Form* in eine Form mit eingeschränkten $RotX_0$ wandeln.

3.11 Girder

3.11.1 PieceCount, X, Y, Z, GirderName, Length, AngleToX

AngleToX gibt die Verlege-Richtung an (ausgehend von der x-Achse).
Die Werte liegen im Bereich]-180°, 180°].

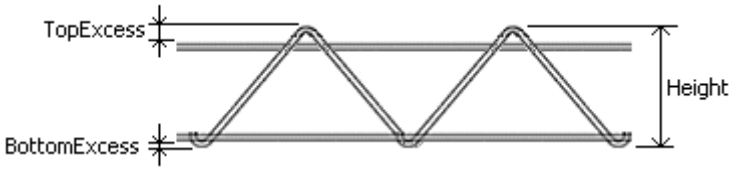
X und *Y* beziehen sich auf die X/Y-Lages des Obergurtes (Kernmaß).

Z bezieht sich auf die Z-Lage der Untergurten (Kernmaß).

3.11.2 NoAutoProd

Gesetzt, wenn der Gitterträger *nicht* automatisch produziert werden soll.

3.11.3 Height, TopExcess, BottomExcess



Die *Height* ist die Z-Differenz vom tiefsten zum höchsten Punkt.

Der *TopExcess* ist der Z-Abstand des höchsten Punktes zur Oberkante des Obergurts.

Der *BottomExcess* ist der Z-Abstand des tiefsten Punktes zur Unterkante des Untergurtes.

3.11.4 Weight, TopFlangeDiameter, BottomFlangeDiameter

TopFlangeDiameter und *BottomFlangeDiameter*, *DiagonalWireDiameter* sind in mm; *Weight* in Kg.

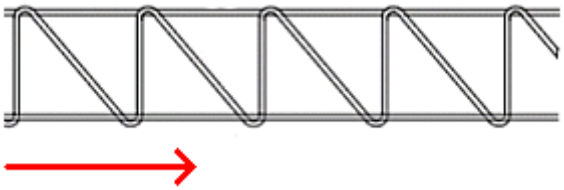
3.11.5 GirderType

GirderType ist eine numerische Typ-Kennung. Es wird empfohlen, folgende Werte-Festlegungen zu verwenden:

0 = Standardträger (ohne zusätzliche Definition).
1 = Allgemeiner Grundbewehrungsträger.
2 = Schubträger.
3 = Zulageträger.
6 = Versaträger
7 = Linksseitiger Versaträger
8 = Rechtsseitiger Versaträger

Entspricht der Einer-Stelle der UNICAM-Trägerart.

Richtung der Schubträger: die Richtung der Schubträger ist so definiert, dass die schrägen flankend abfallend sind:



3.11.6 MountingType

Einbau-Anweisung:

0 = Keine Angabe.

1 = Manuell zu befestigen (Nacharbeit nötig).

2 = Manuell einzulegen.

Entspricht der Zehner-Stelle der UNICAM Trägerart.

3.11.7 ArticleNo

Artikelbezeichnung des Gitterträgers.

3.11.8 Machine

Text-Feld für Angabe der Produktions-Maschine.

Eine maschineninterne Produktionsliste kann optional auch als eigene Maschine behandelt werden. In diesem Fall wird empfohlen, folgendes Format zu wählen:

GTA:2

In diesem Beispiel ist "GTA" die eigentliche Maschinenangabe, und "2" die Nummer der Produktionsliste.

3.11.9 Period, PeriodOffset

Im Feld *Period* kann die Gitterträger-Periode angegeben werden (in mm). Typisch ist hier ein Wert von 200. Falls der Wert nicht angegeben wird, oder 0 ist, ist ein applikationsabhängiger Default-Wert anzunehmen.

Das Feld *PeriodOffset* gibt an, in welchem Abstand vom Gitterträger-Anfang der erste Perioden-Tiefpunkt liegt. Beim nicht geschnittenen Gitterträger ist dieser Wert normalerweise 0; beim gerastert geschnittenen Gitterträger ist dieser Wert 0 oder gleich der halben Periodenlänge; beim beliebig geschnittenen Gitterträger sind alle Werte möglich, positive, wie auch negative.

Export nach UNICAM: Die Felder *Period* und *PeriodOffset* werden beim UNICAM-Export in die 2 Reserve-Felder geschrieben, die sich am Beginn der Zeile 5 des BRGIRDER-Blocks befinden.

3.11.10 Width

Im Feld *Width* kann die Gitterträger-Breite angegeben werden (in mm); gemessen wird ganz außen am Gitterträger. Typisch ist hier ein Wert von 80. Falls der Wert nicht angegeben wird, oder 0 ist, ist ein applikationsabhängiger Default-Wert anzunehmen.

3.11.11 AnchorBar

Verankerungsstäbe.

Type: Typ-Kennung.

Length: Länge in mm.

Position: Verlegeposition in mm.

3.11.12 GirderExt

Generische Untertabelle des Gitterträgers.

Die genaue Bedeutung der Felder ist abhängig vom Typ.

Für alle Typen gibt es das **Position**-Feld: dieses beschreibt eine Position in mm, gemessen vom Gitterträger-Anfang.

Das **Flags**-Feld enthält 32 Bits, die typspezifisch und/oder applikationsspezifisch verwendet werden können. Die Bedeutung der Felder **Val0, …, Val3** ist ebenfalls typspezifisch und/oderapplikationsspezifisch.

3.11.12.1 Type = splicePos

Spleißpunkt an dem der Gitterträger bei der Gitterträger-Position gestückelt wurde.

- Val0: Phasensprung in mm am Schweißpunkt. Entspricht der Gitterträger-Länge, die am Spleißpunkt 'ausgelassen' wurde. Der Wert kann auch negativ sein.
- Val1: Überlappungslänge am Spleißpunkt. Entlang dieser Strecke hat man einen doppelten Gitterträger. Der Wert kann auch negativ sein.

3.11.12.2 Type = FixingPos

Punkt an dem der Gitterträger auf der Bewehrung fixiert wird (Verschweißt oder geheftet).

Die Werte Val0, …, Val03 sind applikationsabhängig. Bei Applikationen, die mit festen Schweiß-Tabellen übergibt man in Val0 das Schweißprogramm. In anderen Applikationen können Schweißstrom, Schweißzeit und Ähnliches angegeben werden.

3.11.12.3 Type = GirderGripPos

Position an der der Gitterträger gegriffen wird um eingelegt zu werden (Griffposition für den Transport des Einzel-Gitterträgers).

3.11.12.4 Type = MeshGripPos

Position an der der Gitterträger gegriffen wird wenn das Element transportiert wird.

3.11.12.5 Type = SupportPos

Position an der ein Auflage-Zwischenstück anzubringen ist.

3.11.12.6 Export nach UNICAM

Die GirderExtRows werden beim Export nach UNICAM als Gitterträgerschweißpunkte eingetragen. Die Schweißleistung enthält die Typ-Information:

- Schweißleistung = **0xx**: *Unknown*.
- Schweißleistung = **1xx**: *SplicePos*.
- Schweißleistung = **2xx**: *FixingPos*.
- Schweißleistung = **3xx**: *GirderGripPos*.
- Schweißleistung = **4xx**: *MeshGripPos*.
- Schweißleistung = **5xx**: *SupportPos*.

Der Wert **'xx'** wird mit ***k = Round(Val0 / 5.0)*** gefüllt. Um sicherzustellen, dass *k* im Bereich 0 bis 100 liegt, wird noch folgende Transformation gemacht: *k* wird auf [-50, 49] beschränkt und bei negativen werten wird 100 hinzuaddiert (d.h. bei einem *k* = -7 wird xx = 93 gesetzt).

3.11.13 Section

Über die *Section*-Tabelle kann das Periodenmuster des Gitterträgers exakt festgelegt werden. Für Eigenschaften, die nicht über *Section*-Einträge definiert werden, gelten die übergeordneten Angaben im *Girder*-Eintrag, bzw. Defaultwerte, falls gar nichts spezifiziert wird.

Ein *Section*-Eintrag beschreibt einen Periodenabschnitt, der immer mit einem Periodentiefpunkt beginnt und mit einem Periodentiefpunkt endet.

3.11.13.1 Felder der Section-Tabelle

- **L**: Länge eines Periodenabschnitts in mm. Typisch sind Werte um 200 mm.
  Der Wert sollte immer positiv sein: $L > 0$.
- **S**: Verschiebung des obersten Diagonaldrahtpunktes in Bezug auf die Mitte des Periodenabschnitts. Für einen normalen Gitterträger ist der Wert 0. Für einen Schubträger liegt der Wert typisch um -100. In jedem Fall sollte der Wert immer in den Grenzen des betreffenden Periodenabschnittes liegen, d.h. $|S| \leq \frac{L}{2}$.
  Der Wert von S beschreibt die genannte Verschiebung aber in einer „idealisierten" Form: wenn V die tatsächliche Verschiebung (in mm) ist, so ist S wie folgt definiert:
  $$S := V \cdot \frac{L}{L - 2 \cdot (r_B + r_T)}, \quad \text{bzw.} \quad V = S \cdot \frac{L - 2 \cdot (r_B + r_T)}{L}$$
  Hierbei sind r_T und r_B die jeweils oberen und unteren Diagonaldrahtbiegeradien. D.h. bei Berücksichtigung der Biegeradien ist die tatsächliche Verschiebung betragsmäßig geringer als der in S angegebene Wert[62].
- **F**: Länge des flachen Teilstückes (i.d.R. nur bei niederen Schubträgern mit doppelt geschweißtem Untergurt). Das flache Teilstück liegt immer auf jener Seite des Periodenabschnittes, auf der der Diagonaldraht flacher verläuft. Wenn also S negativ ist, liegt das flache Teilstück auf der positiven Seite des Abschnittes (und umgekehrt). Zudem soll die Einschränkung $0 \leq F \leq 2 \cdot |V|$ erfüllt sein[63].

3.11.13.2 Zuordnung der Section-Angaben

Die Zuordnung der *Section*-Einträge zu den effektiven Gitterträger-Abschnitten erfolgt nach folgenden Regeln:

a) Die *Section*-Einträge beschreiben ein sich wiederholendes Muster. Wenn beispielsweise die 3 *L*-Angaben 195, 195, 200 gemacht werden, bedeutet das, dass die ersten beiden Perioden eine Länge von 195 mm haben, und die dritte Periode eine Länge von 200 mm hat. Die vierte und fünfte Periode haben wieder die Länge 195 mm, usw.

b) Die *Section*-Angaben beschreiben nur ein Muster; sie sagen nichts über die tatsächliche Länge des Gitterträgers aus. Diese wird im *Length*-Feld des *Girder*-Eintrags festgelegt.

c) Das Periodenmuster beginnt normalerweise beim Gitterträgeranfang. Wenn aber ein von 0 verschiedener *PeriodOffset*-Wert angegeben wird, ist der Beginn des Musters entsprechend verschoben[64].

3.11.13.3 Berechnungen zur Gitterträger-Form

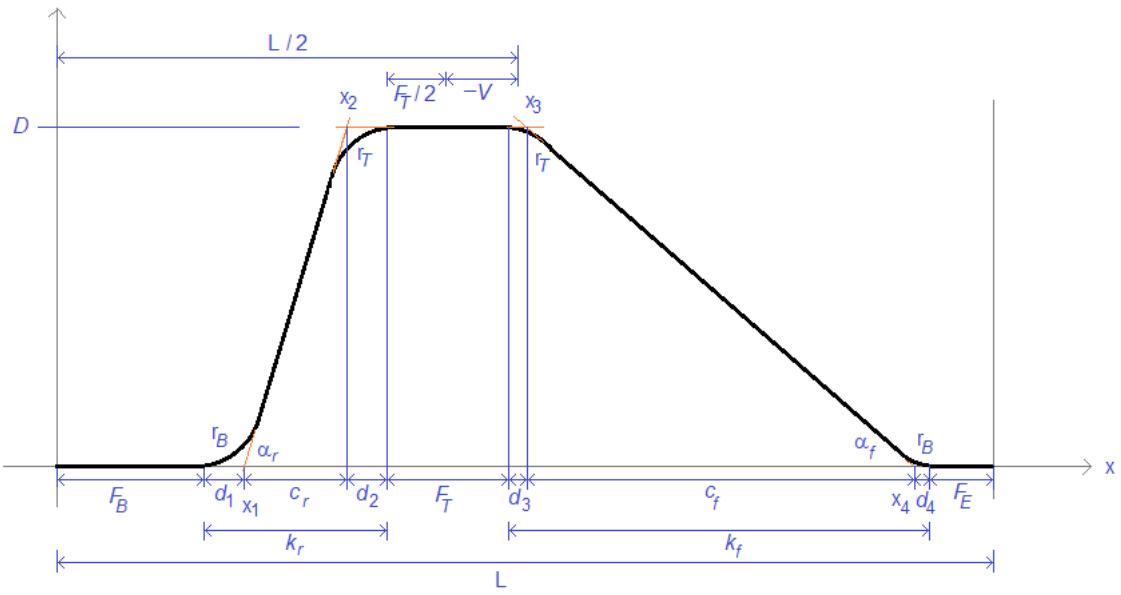
Es soll hier der Diagonaldraht in aufgestellter Lage betrachtet werden:

---

[62] Diese „idealisierte" Definition für *S* hat folgenden Grund: die tatsächliche Verschiebung *V* ist von den Biegeradien abhängig. Diese sind aber unter Umständen nicht bekannt, wodurch es von Vorteil ist, den Wert *S* so zu definieren, dass er typischerweise unabhängig von den Biegeradien ist. Konkret führt ein Wert von *S=L/2* immer zu einem ganz vertikal verlaufenden Diagonaldrahtabschnitt, unabhängig von den Biegeradien.

[63] Das flache Teilstück soll verhindern, dass bei niederen Schubträgern der Diagonaldraht zu flach verläuft. Das flache Teilstück ist daher immer auf jener Seite, auf der der Diagonaldraht flacher verläuft.
Mit der Einschränkung $F \leq 2 \cdot |V|$ verhindert man einerseits, dass *F* die Schubrichtung invertiert, und schließt andererseits den Fall den Fall *F>0* bei *S=0* aus; in diesem Fall wäre die Lage des Flachen Teilstückes unbestimmt. Der Grenzfall $F = 2 \cdot |V|$ beschreibt einen Diagonaldraht mit symmetrischer Form (steigende und fallende Flanke gleich steil).

[64] Beachte: Bei einem positiven Wert von *PeriodOffset* wird der Start des Periodenmusters in den Gitterträger hinein verschoben. Der erste Teil des Gitterträgers liegt dann vor dem Beginn des Periodenmusters. Dieser Teil hat aber dennoch eine definierte Struktur, da sich das Muster periodisch wiederholt, und somit auch vor seinem Startpunkt definiert ist (Vor dem Beginn des Musters reiht sich das das Ende des selben Musters, eben weil das Muster in unendlicher Folge wiederholt wird).



Aufgrund der PXML-Daten sind folgende Größen bekannt[65]:

F_B, F_T, F_E, L, V, r_B, r_T, D

Mit diesen Angaben lassen sich alle anderen Größen ermitteln:

$$k_r = \frac{L}{2} + V - \frac{F_T}{2} - F_B, \qquad k_f = \frac{L}{2} - V - \frac{F_T}{2} - F_E$$

$$d_1 = r_B \cdot \tan\frac{\alpha_r}{2}, \qquad d_2 = r_T \cdot \tan\frac{\alpha_r}{2}, \qquad d_3 = r_T \cdot \tan\frac{\alpha_f}{2}, \qquad d_4 = r_B \cdot \tan\frac{\alpha_f}{2}$$

$$c_r = \frac{D}{\tan\alpha_r} = \frac{D \cdot (1 - \tan^2\frac{\alpha_r}{2})}{2\tan\frac{\alpha_r}{2}}, \qquad c_f = \frac{D}{\tan\alpha_f} = \frac{D \cdot (1 - \tan^2\frac{\alpha_f}{2})}{2\tan\frac{\alpha_f}{2}}$$

Mit $t_r := \tan\frac{\alpha_r}{2}$ und $t_f := \tan\frac{\alpha_f}{2}$ hat man:

$$d_1 = r_B \cdot t_r, \qquad d_2 = r_T \cdot \mathrm{t}_r, \qquad d_3 = r_T \cdot \mathrm{t}_f, \qquad d_4 = r_B \cdot \mathrm{t}_f$$

$$c_r = \frac{D \cdot (1 - \mathrm{t}_r^2)}{2 \cdot \mathrm{t}_r}, \qquad c_f = \frac{D \cdot (1 - \mathrm{t}_f^2)}{2 \cdot \mathrm{t}_f}$$

Wegen

$$k_r = d_1 + d_2 + c_r, \qquad k_f = d_3 + d_4 + c_f$$

ergibt sich

$$k = g \cdot t + \frac{D \cdot (1 - \mathrm{t}^2)}{2 \cdot \mathrm{t}}, \quad g := r_B + r_T$$

(Hierbei ist $k := k_r$ und $t := t_r$ oder $k := k_f$ und $t := t_f$)

Das kann schließlich nach t aufgelöst werden:

$$t = \frac{\sqrt{D^2 - 2Dg + k^2} - k}{D - 2g}$$

[65] Nach aktueller Definition ist F_T stets 0 und von F_B und F_E ist jeweils höchstens einer ungleich 0. In der nachstehenden Berechnung wird aber der etwas allgemeinere Fall von beliebigen nichtnegativen F_T, F_B, F_E betrachtet. r_B und r_T sind nicht direkt gegeben, werden aber typischerweise zu $\frac{5W}{2}$ angenommen, wobei W hier der Drahtdurchmesser des Diagonaldrahtes ist.

3.12 Alloc

Ein *Alloc*-Block beschreibt ein Staffelungsmuster für Eisen oder Gitterträger.

Verwendet wird dies vor allem für die Beschreibung von *Körben*: der *Alloc*-Block beschreibt dort das Verlegungsmuster der Bügel.

Type-Attribut

Es gibt folgende Typen:

- **Bar:** Der *Alloc*-Block bezieht sich auf Eisen (Bügel, Staffeleisen).
- **Girder:** Der *Alloc*-Block bezieht sich auf Gitterträger.

3.12.1 GuidingBar (Leiteisen)

Falls im Feld *GuidingBar* ein gültiger Eisen-Index steht, erfolgt die Staffelung entlang dieses Eisens. Die *Region*-Werte definieren dann Staffelpositionen auf diesem Leiteisen, d.h. die "geleiteten" Eisen werden entlang des Leiteisensan diesen Staffelpositionen angeordnet.

Begriffsdefinition:

- **Leiteisen (guiding bar):** jenes Eisen, das den Pfad für die Staffelung vorgibt.
- **Staffelpositionen (guiding positions):** jene Punkte auf dem Leiteisen, an denen die geleiteten Eisen angeordnet werden.
- **Geleitetes Eisen (guided bar):** jenes Eisen, das vom Leiteisen "geführt" werden soll.
- **Gestaffelte Eisen (allocated bars):** jene Eisen, die an den realen Positionen liegen, die sich aus der der Staffelung ergeben.

Es gelten folgende Regeln:

a) Der vom Leiteisen definierte Pfad verläuft im Eisenkern des Leiteisens und hat bei Biegungen den Biegeradius 0.

b) Für die Ermittlung der Koordinaten der gestaffelten Eisen, addiert man die *relativen Staffelpositions-Koordinaten* zu den Eisenkoordinaten. Die *relativen Staffelpositionen* sind hierbei die Staffel-Positionen auf dem Leiteisen, bezogen auf den Anfang des Leiteisens. Das originale (ungestaffelte) geleitete Eisen ist also am Startpunkt des Leiteisens zu positionieren.

c) Wenn das Leiteisen Biegungen hat, werden die gestaffelten Eisen zusätzlich gedreht. Gedreht wird nach der Verschiebung, und zwar um den Staffelpositionspunkt. Für jede Leiteisenbiegung, bis zur betreffenden Staffelposition, erfolgt eine Drehung um die Achse der Biegung[66].

3.12.2 Richtungsbestimmung ohne GuidingBar

Falls es keine GuidingBar gibt, wird die Richtung durch das gesteffelte Objekt selbst bestimmt:

Das Eisen bestimmt den Start-Punkt über die entsprechenden X/Y/Z-Koordinaten des *Bar*-Items. Die Richtung wird wie folg festgelegt:

---

[66] Das erste *BendY* des Leiteisens ist hierbei *nicht* als Biegung aufzufassen; d.h. es geht nur um die Biegungen innerhalb des Eisens.
An dieser Stelle mag es verwundern, dass *RotZ* und *BendY0* des Leiteisens nicht in die gestaffelten Eisen eingerechnet werden. Das hat zum einen praktische Gründe, da es recht unanschaulich wäre, wenn das*RotZ* des geleiteten Eisens nicht absolut, sondern relativ zum Leiteisen wäre. Tatsächlich gibt es aber recht fundamentale Gründe, die verhindern *RotZ* und *BendY0* des Leiteisens in die gestaffelten Eisen einzurechnen: *RotZ* und *BendY0* sind für ein gegebenes Leiteisen nicht eindeutig, da es unterschiedliche Kombinationen dieser Werte gibt, die für das Leiteisen äquivalent sind, die aber für die gestaffelten Eisen, wenn sie einberechnet würden, unterschiedliche geometrische Verhältnisse hervorrufen würden.

1) Koordinatensystem um *RotZ* drehen (um die positive z-Achse).
2) Koordinatensystem um $RotX_0$ drehen (Um die positive x-Achse).
3) Koordinatensystem um $BendY_0$ drehen (Um die negative y-Achse).
4) Koordinatensystem um $RotX_1$ drehen (Um die positive x-Achse); dies nur dann, wenn mehr als ein Segment existiert.
5) Die negative y-Achse zeigt dann in die Richtung von positiven Pitch-Werten. Das ist die Achse, über die die erste Biegung definiert wird.

Beachte: in UNICAM ist der Pitch-Wert für Längs- und Quereisen unterschiedlich definiert[67].

- Für Eisen in +X-Richtung (0°): UnicamPitch = – PxmlPitch
- Für Eisen in –X-Richtung (180°): UnicamPitch = PxmlPitch
- Für Eisen in +Y-Richtung (90°): UnicamPitch = PxmlPitch
- Für Eisen in –Y-Richtung (-90°): UnicamPitch = – PxmlPitch
- Für alle anderen Winkel: in UNICAM müssen Einzeleisen angegeben werden.

Die UNICAM-Definition wirkt etwas vertrauter, kann aber nicht konsistent für allgemeine Winkel definiert werden.

Für **Gitterträger** gilt analoges: der Startpunkt wird über die Gitterträger-Koordinaten festgelegt: die Richtung des Abstandes ist wie folgt definiert:

1) Koordinatensystem um AngleToX drehen (um die positive z-Achse).
2) Die negative y-Achse zeigt dann in die Richtung von positiven Pitch-Werten. Das ist die Achse, über die die erste Biegung definiert wird.

Wieder besteht der oben genannte Unterschied zur UNICAM-Definition.

3.12.3 Region

Das Verlegungsmuster besteht aus einer Folge von *Regions*: jede *Region* beinhaltet folgende Informationen:

- **IntervalCount:** Anzahl der Abstände (Intervalle).
  Der Wert ist ganzzahlig und nichtnegativ.
- **Pitch:** Intervallbreite in mm. Der Wert kann auch nicht-ganzzahlig sein, und darf auch negativ sein.
- **IncludeBegin:** Falls gesetzt, gehört der Anfang des Intervalls (und das dort liegende Eisen) zur Region dazu.
- **IncludeEnd:** Falls gesetzt, gehört das Ende des Intervalls zur Region dazu.
- **RefIndex:** Integer-Wert, der das referenzierte Item (Eisen oder Gitterträger) bestimmt. Ein Wert von 0 bedeutet, dass das erste Eisen (oder der erste Gitterträger) des betreffenden *Steel*-Blocks referenziert wird.
  Falls der *RefIndex* NULL, negativ, oder zu groß ist, gilt er als nicht zugewiesen. Die *Region* definiert dann einen Abstandsoffset ohne Zuordnung eines Eisens.

Anmerkungen:

i) Die **Länge** der *Region* ergibt sich aus **L = IntervalCount * Pitch**.
ii) Eine *Region* definiert Positionen
Die Position *k* der *m*-ten Region wird wie folgt berechnet:
$P_{m,k} = \sum_{i=0}^{m-1} L_i + k \cdot Pitch_m\,, \quad L_i = IntervalCount_i \cdot Pitch_i.$
Die so berechnete Position ist zunächst nur ein skalarer (eindimensionaler) Wert. Über die

[67] Siehe z.B. Unitechnik-Schnittstellen-Definition 6.0, 4.9.5, Punkt 10: "Positive Teilung bedeutet positive Koordinatenrichtung, …, Die Verwendung der Teilung ist NUR für Winkel von 0°, 90°, 180° und 270° erlaubt."

Verfahren aus den Abschnitten 3.12.1 und 3.12.2 kann daraus ein konkreter Raum-Punkt bestimmt werden.

iii) Wenn die *Region* kein gültiges *RefIndex* hat, sind ihr keine Bügel zugeordnet werden[68].

iv) Wenn die *Region* ein gültiges *RefIndex* hat, werden mindestens *IntervalCount-1* Bügel zugeordnet. Wenn *IncludeBegin* gesetzt ist, kommt ein weiter Bügel hinzu. Wenn *IncludeEnd* gesetzt ist, kommt ebenfalls ein Bügel hinzu. Maximal können der *Region* somit *IntervalCount+1* Bügel zugeordnet werden.

v) Üblicherweise wird man *IncludeBegin* einer Region und *IncludeEnd* der vorhergehenden Region synchronisieren. Wirklich frei ist man daher nur am Anfang der ersten und am Ende der letzten *Region*. Diese Synchronisierung ist jedoch nicht zwingend vorgeschrieben.

vi) Aufgrund des oben stehenden Musters ergibt ein *Alloc*-Block für einen darin referenzierten Bügel eine Stückzahl. Diese als **AllocCount** zu bezeichnende Stückzahl wird also wie folgt berechnet:

$$AllocCount = \sum_{i=0}^{RegionCount-1} D_i \cdot (IntervalCount_i - 1 + E_i + B_i), \quad AllocCount >= 0$$

$B_i = 1$ falls $IncludeBegin$ gesetzt
$E_i = 1$ falls $IncludeEnd$ gesetzt
$D_i = 1$ falls der betreffende Bügel in Region_i vorkommt, andernfalls 0

Beachte: *AllocCount*-Wert ist per definitionem nichtnegativ, obwohl er rein rechnerisch bei *IntervalCount=0* negativ sein könnte.

Beachte: ein *Alloc*-Block definiert für jeden Bügel, der im *Alloc*-Block referenziert wird, einen eigenen *AllocCount*-Wert.

vii) Die Summe der *AllocCounts* zu einem Bügel gibt an, wie viele Bügel eine zugewiesene Staffelposition haben. Üblicherweise wird diese Anzahl mit der Stückzahl des Bügels übereinstimmen (*Bar.PieceCount*). Diese Übereinstimmung ist aber nicht zwingend vorgeschrieben, d.h. *Bar.PieceCount* kann überzählige oder fehlende Bügel aufweisen.
Falls es **überzählige Bügel** gibt, werden diese ihrer originalen X/Y/Z-Position zugeordnet; diese überzähligen Bügel werden also getrennt von den *Alloc*-Blöcken betrachtet.
Falls es **fehlende Bügel** gibt, werden die letzten Bügel der letzten *Alloc*-Blöcke als fehlend betrachtet.

viii) Eine *Region* mit ***Pitch=0*** kann rechnerisch behandelt werden wie jede andere *Region*. Verwendet wird dieser Sonderfall üblicherweise um Bügel darzustellen, die nicht verschweißt werden, sondern lose mitgeliefert werden (typischerweise am vordere und am hinteren Ende eines Korbes).

ix) Die Definition der *Alloc*- und *Region*-Datenstrukturen lässt es zu, dass ein Eisen in mehreren *Regions* referenziert wird, die auch in unterschiedlichen *Alloc*-Blöcken liegen können. Üblicherweise wird ein Eisen aber nur in *Region*-Blöcken referenziert, die zu einem einzigen *Alloc*-Blöck gehören. Das Eisen hat also üblicherweise höchstens ein Staffelmuster (=*Alloc*-Block). Umgekehrt hat ein Staffelmuster oft mehrere unterschiedliche Eisen.

3.13 SteelExt

Zusatz-Einträge zum Steel-Block. Die Art der Einträge ist applikationsabhängig und wird über das *Type*-Attribut unterschieden. Die Bedeutung des *Info*-Feldes ist abhängig vom *Type*-Attribut. Zusätzlich wird es üblicherweise weitere applikationsspezifische Felder geben (I_P_-Tags).

[68] Der Einfachheit halber wird hier nur von *Bügeln* gesprochen. Analoges gilt aber auch für andere Staffeleisen und auch für Gitterträger.

3.14 Feedback

Die **Feedback**-Tabelle unterscheidet sich ganz wesentlich von den anderen PXML-Tabellen: die *Feedback*-Tabelle dient nicht der Beschreibung von Produktionsdaten, sondern beinhaltet Maschinen-Rückmeldung. Wie im weiter unten noch näher erläutert, gibt es 2 Arten von Rückmeldungen:

- *PTS*-Meldung: informiert vorab über die Produzierbarkeit bestimmter Items.
- *Machine-Return*: informiert über die erfolgte Produktion.

Beim Datentransfer vom CAD zur Maschine wird es üblicherweise keine *Feedback*-Tabelle geben. Beider Rückmeldung der Maschine (oder des Test-Systems) an das CAD wird es häufig nur die *Feedback*-Tabelle geben, und keine weiteren Daten; in speziellen Fällen können aber auch andere PXML-Daten von der Maschine an das CAD- oder Leitsystem zurückgemeldet werden.

3.14.1 Production Test Service (PTS)

Der **Production Test Service** (kurz **PTS**) erlaubt es einem CAD- oder Leitsystem eine Anfrage beim Produktionssystem zu stellen, um die Produzierbarkeit eines Produktes abzufragen. Die Datenerstellung kann dann unter direkter Einbeziehung der Maschinen-Möglichkeiten erfolgen; man erreicht damit eine große Prozesssicherheit und eine große Optimalität des Produktionsablaufs.

Der Bedarf an *PTS*-Systemen ist vor allem im Zusammenhang modernen Mattenbiegeanlagen aufgekommen. Diese Anlagen haben einen immer weiter wachsenden Leistungsumfang, der aber, mit zunehmender technischer Komplexität, nicht mehr in Form von einigen Grenzwerten beschrieben werden kann. Die vollautomatische Produzierbarkeit eines komplexen Bewehrungskorbes hängt von so vielen Details ab, dass sie letztlich nur unter direkter Einbeziehung der Maschinensoftware zuverlässig geprüft werden kann; und genau das soll mit *PTS* erreicht werden.

PTS besteht aus 2 Teilen:

- **PTS-Server:** ist eine Service-Anwendung, die vom Maschinenhersteller bereitgestellt werden muss; dieser Service läuft auf den CAD-Workstations oder irgendwo zentral im Netz. Der *PTS*-Server wartet auf Anfragen und beantwortet diese; er muss dazu die Berechnungslogik und die Parameter der Maschinensoftware berücksichtigen.
- **PTS-Client:** ist vom CAD-Hersteller in der CAD-Anwendung bereitzustellen (und/oder im Leitsystem). Mittels *PTS*-Client werden aus der CAD-Anwendung heraus Anfragen an den *PTS*-Server gestellt. Die Server-Antwort muss dann im CAD-System visualisiert werden.

Die Zuordnung einer Antwort zu einer vorher gesendeten Anforderung erfolgt über die *GlobalID* der *DocInfo*-Tabelle. Diese ID sollte daher für jede Anforderung individuell gesetzt werden.

Die **Anfrage (Request)** des *PTS*-Client an den PTS-Server erfolgt in Form eines PXML-Dokumentes, das Produktionsdaten enthält, also *Order*-Blöcke mit den zugehörigen Child-Strukturen.

Die **Antwort (Feedback)** des *PTS*-Servers erfolgt in Form eines PXML-Dokumentes, das *Feedback*-Blöcke enthält. Wenn der *PTS*-Server Produktionsdaten von sich aus verändert, kann er zusätzlich zu den *Feedback*-Blöcken auch *Order*-Blöcke zurückmelden, um so die veränderten Daten zu beschreiben[69]. Die Rückmeldung der *Order*-Blocke (=Produktionsdaten) ist aber als optionale Zusatzfunktion anzusehen; das zentrale Element der der Rückmeldung sind die *Feedback*-Blöcke.

[69] Die zurückgemeldeten Produktionsdaten müssen natürlich wieder über die *GlobalID* auf die Originaldaten Bezug nehmen. Fall ein Item der Originaldaten in der Rückmeldung mehrfach referenziert wird (weil beispielsweise ein originales Eisen in mehrere Positionen aufgetrennt werden musste), muss in allen betreffenden Rückmeldungs-Items die selbe*GlobalID* angegeben werden.

3.14.2 Machine Return

MachineReturn ist eine Funktionalität, die eng verwandt mit der *PTS*-Rückmeldung ist. Im Gegensatz zum *PTS* geht es hierbei aber nicht um eine Test-Rückmeldung, sondern um eine Produktionsrückmeldung der Produktionsmaschinen an das übergeordnete Leitsystem. Im Wesentlichen werden hierbei produzierte Stückzahlen zurückgemeldet.

Falls die Produktionsmaschine eigenständig Produktionsdaten verändert, kann es sinnvoll sein, in der Rückmeldung auch die veränderten Produktionsdaten anzugeben (*Order*-Blöcke samt Child-Struktur). Diese (optionale) Funktionalität kann sogar so weit gehen, dass Daten, die an der Maschine manuell eingegeben werden, auf diesem Wege in das übergeordnete Leitsystem zurückgepflegt werden.

Eine weitere Form von Produktionsrückmeldungen gibt es vom Leitsystem zu übergeordneten ERP-System.

3.14.3 Felder des Feedback-Blocks

3.14.3.1 Feedback-Attribute

1) **ItemType**: Gibt den Objekt-Typ an, auf den sich die Rückmeldung bezieht. Typischerweise steht hier ein PXML-Tabellenname.
2) **GlobalID**:Gibt an, auf welches Produktionsdaten-Item sich die Rückmeldung bezieht. Um einen *Feedback*-Eintrag eindeutig zuordnen zu können, benötigt man sowohl den *ItemType* als auch die *GlobalID*. Nur beide zusammen identifizieren das betroffenen Produktions-Item.

Beide genannte Attribute sollen pro *Feedback*-Block genau einmal angegeben werden.

3.14.3.2 Feedback-Felder

1) **MessageType**: Definiert den Typ der Meldung. Folgende Werte sind möglich:
   a. **info**: Reine Information, ohne Warnungscharakter. "info" ist auch der Default-*MessageType*, der angenommen wird, wenn es keine Angabe gibt.
   b. **hint**: Hinweis mit kleiner Warn-Stufe. Das wird beispielsweise für Items verwendet, die sich problemlos produzieren lassen, für die aber auf fehlende Optimierung in Bezug auf Produktionsgeschwindigkeit hingewiesen werden soll.
   c. **warning**: Warnung mittlerer Stufe. Das Item kann zwar produziert werden, aber nur unter Schwierigkeiten; eventuell wird es in minderer Qualität oder unter großer Belastung der Maschine produziert.
   d. **error**: Fehler. Das Item kann so nicht produziert werden (Das Item wird entweder gar nicht produziert, oder nur in stark abgeänderter Form).
   e. **program**: Rückmeldung für automatische Auswertung durch den PTS-Client. Ist primär nicht für die Benutzer-Anzeige gedacht. Hat typischerweise keinen Description-Text.
2) **Code**: Alphanumerischer Code, der die Bedeutung der *Feedback*-Meldung identifiziert. Die Liste von Codes mit der ihnen zugeordneten Bedeutung wird vom Maschinenhersteller bereitgestellt.Bei *PTS*-Meldungen wird empfohlen, über das *Code*-Feld eine eindeutige Fehler-Kodes anzugeben. Im Rahmen von *Machine-Return*-Meldungen kann der *Code* benutzt werden, um Rückmeldungs-Events zu identifizieren. Typisch sind hier Codes wie *imported* (datenmäßig importiert), *started* (Produktion gestartet), *produced* (als Einzelposition produziert; bei Betonelementen: betoniert), *embedded* (eingebaut, verschweißt; bei Betonelementen: eingelagert), *delivered* (ausgeliefert, übergeben an das Folgesystem; bei Betonelementen: ausgelagert/abgehoben).

Für die MES→ERP-Rückmeldung gibt es eine standardisierte Ereignis- und Code-Liste: siehe hierzu die **PXML Web API** Beschreibung (siehe www.pxml.eu).

3) **InfoValue**: Zusatz-Information zur *Feedback*-Meldung. Hier kann ein numerischer oder ein alphanumerischer Wert angegeben werden. Die genaue Bedeutung dieses Feldes hängt vom *Code* ab.
4) **PieceCount**: Stückzahl (Integer-Wert); gibt an, wie viele Stück vom Angegebenen Item zurückgemeldet werden. Typischerweise steht hier 1.
   Dieses Feld wird für *Machine-Return*-Meldungen verwendet, und fehlt in *PTS*-Meldungen normalerweise.
5) **MaterialType**: Typ des verwendeten Materials.
   Bei *Bar*-Items wird hier typischerweise der Drahtdurchmesser eingetragen; falls es zu einem Durchmesser verschiedene Draht-Typen gibt, wird hier eine allgemeinere alphanumerische Draht-ID eingetragen.
   Bei Gitterträgern wird typischerweise der Träger-Typ eingetragen.
   Dieses Feld wird für *Machine-Return*-Meldungen verwendet, und fehlt in *PTS*-Meldungen normalerweise.
6) **MaterialBatch**: Chargenkennung des eingesetzten Materials.
   Für Rundstahl vom Coil wird folgendes Format empfohlen: **"12345@AR177228C"**.
   Hierbei ist "12345" eine ID, die das Coil eindeutig identifiziert; "AR177228C" ist hier die Stahl-Charge. Da die Coil-Charge aus der ID abgeleitet werden kann, ist es häufig sinnvoll, nur die ID anzugeben (also nur "12345"); wenn keine ID bekannt ist, kann man auch nur die Charge angeben, angeführt vom Trennzeichen '@' (also "@AR177228C").
   Dieses Feld wird für *Machine-Return*-Meldungen verwendet, und fehlt in *PTS*-Meldungen normalerweise.
7) **MaterialWeight**: Gewicht des eingesetzten Materials in kg (Gesamtgewicht für alle produzierten Einheiten).
   Dieses Feld wird für *Machine-Return*-Meldungen verwendet, und fehlt in *PTS*-Meldungen normalerweise.
8) **ProdDate**: Produktions-Zeitstempel (Ende-Zeitpunkt der Produktion). Wird typischerweise in *Machine-Return*-Meldungen angegeben, um den genauen Produktionszeitpunkt festzuhalten.
9) **Machine**: Identifikation der Maschine oder des PTS-Servers, der die Meldung generiert. Dieses Feld ist insbesondere dann bedeutsam, wenn mehrere PTS-Server hintereinandergeschaltet sind. (Falls innerhalb einer physischen Maschine zwischen unterschiedlichen Produktionslisten unterschieden werden soll, kann, wie an anderer Stelle die Schreibweise MMM:X empfohlen werden, wobei MMM die physische Maschine identifiziert und X die Produktionsliste.)
10) **Description**: Optionale Texteinträge, um die Meldung in Textform zu beschreiben. Es können mehrere Sprachen angegeben werden. Hierbei ist für das *Culture*-Attribut mittels ISO 639 Language Codes anzugeben, optional erweitert durch die ISO 3166 Country Codes; man hat also Angaben der Form "en" oder "en-US".
    Der Description-Text kann auch lang und mehrzeilig sein. Neben der eigentlichen Fehlermeldung kann es auch Korrekturempfehlungen geben, oder Angaben über automatisch durchgeführte Korrekturen.
    Beispiel: "Rastermaß Y=50mm nicht eingehalten. Eisen in Y um -25mm verschoben."
    Ein *Feedback*-Eintrag sollte pro Sprache nur einen *Description*-Eintrag haben.

In *Machine-Return*-Meldungen, in denen lediglich produzierte Stückzahlen gemeldet werden, wird man den *MessageType* nicht angeben, oder ihn auf "info" setzen. Im Prinzip kann aber eine Produktionsmeldung auch eine Warnung miteinschließen, so dass eine *Machine-Return*-Meldung auch den anderen *MessageType*s kombiniert werden kann.

3.14.4 FbVal

Bei einer *Feedback*-Meldung können über *FbVal*-Einträge zusätzliche Werte zurückgemeldet werden. Jeder *FbVal*-Eintrag hat zwei Attribute: eine Typ *T* und einen Wert *V*.
Die folgenden Tabellen beschreiben die *FbVal*-Typen und -Werte, die vom Standard definiert sind.

T	Beschreibung	Beispiel
OrdNo	Auftragsnummer	AA00048386
Customer	Kunde	XyConstruction
ElemNo	Elementnummer	5522A
PartType	Elementteilangabe bei (wird typischerweise nur bei DW/TW verwendet)	01
Pos	Positionsbezeichnung	17B
ShiftID	Identifikation der Produktionsschicht auf die sich eine Produktionsrückmeldung bezieht	MorningShift
ShiftStart	Startzeitpunk der Produktionsschicht auf die sich eine Produktionsrückmeldung bezieht. Der Schichtstart ist für Auswertungen unter anderem deshalb interessant, weil er manchmal noch in den Vortag fällt, also nicht unbedingt vom Datum her mit dem *ProdDate* übereinstimmt.	2017-02-20T16:32:33+01:00
PalDataID	Identifikation der logischen Fertigungseinheit	123456
PalNo	Identifikation der physischen Produktionseinheit	43
Size	Nominale Abmessung, wie vom CAD angefordert. typischerweise nominale Länge in mm. Bei Matten: typischerweise X- und Y-Abmessung in der Form 5740x1320. Die nominale Abmessung soll zusammen mit der nominalen Typangabe hinreichend viele Informationen für die Lager-Zu-Buchung liefern.	600
ProdSize	Nominale Abmessung, wie effektiv produziert. Dieses Feld unterscheidet sich vom *Size*-Feld dann, wenn die Maschine die Soll-Daten verändert, um nicht produzierbare Soll-Abmessungen auf Abmessungen zu korrigieren, die für die Maschine zulässig sind.	800
NomTy	Angeforderter Produkttyp (aus CAD-Daten). Bei „Bar“-Einträgen kann das z.B. eine Biegeform-Bezeichnung sein. Bei „Girder“-Einträgen ist es typischerweise der Gitterträger-Typ.	M10
ProdTy	Effektiv produzierter Produkttyp. Kann von NomTy abweichen. Bei „Girder“-Einträgen wird typischerweise die in der Maschine parametrierte Gitterträger-Typbezeichnung eingetragen (dieser ist in der	B5

	Maschine typischerweise über eine Alias-Tabelle mit NomTy verknüpft). Bei „Bar“-Einträgen kann hier eine Biegeform-Bezeichnung stehen, oder einfach die Anzahl der Biegungen in der Form „B5“ für ein Eisen mit 5 Biegungen.	
ProdArticle	ERP-Artikel-Code des produzierten Teils.	AXL554
Total_Kg	Reales Gesamtgewicht in Kg des effektiv produzierten Objektes.	9.123
Wr	Drahteigenschaften zu einem Eisen (siehe Drahtinformation); Bei „Bar“-Einträgen ist es das Eisen selbst. Bei „Steel“-Einträgen ist es entweder ein Einzeleisen oder mehrere Eisen desselben Typs zusammengefasst. Dieser Eintrag kann auch mehrmals vorhanden sein.	D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=7.756 Art=GD12345
Wr_Tp	Drahteigenschaften für den Obergurt (siehe Drahtinformation)	D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=7.756
Wr_B1	Drahteigenschaften für den Untergurt links (siehe Drahtinformation)	D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=7.756
Wr_B2	Drahteigenschaften für den Untergurt rechts (siehe Drahtinformation)	D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=7.756
Wr_D1	Drahteigenschaften für den Diagonaldraht links (siehe Drahtinformation)	D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=7.756
Wr_D2	Drahteigenschaften für den Diagonaldraht rechts (siehe Drahtinformation)	D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=7.756
Wld	Informationen zu einem generischen - Schweißpunkt (siehe Schweißpunktinformation0)	X=2663.4 I=9774
Wld_Tp	Informationen zu einem Obergurt-Schweißpunkt (siehe Schweißpunktinformation)	P=7322 T=226 X=2563.4 I=9664
Wld_B1	Informationen zu einem Untergurt links-Schweißpunkt (siehe Schweißpunktinformation0)	X=2663.4 I=9774
Wld_B2	Informationen zu einem Untergurt rechts-Schweißpunkt (siehe Schweißpunktinformation)	I=9333 T=230 X=2560.4
Wld_Target	Sollwerte für generische-Schweißpunkte (siehe Schweißpunktinformation) Wenn hier keine X/Y-Position angegeben wird, so gelten die Werte für das gesamte Objekt.	T=300 I=9800
Wld_Tp_Target	Sollwerte der Obergurt-Schweißpunkte (siehe Schweißpunktinformation)	T=350 I=10200

	Wenn hier kein Abstand zum Anfang angegeben wird, so gelten die Werte für den gesamten Gitterträger.	
Wld_B1_Target	Sollwerte der Untergurt links-Schweißpunkte (siehe Schweißpunktinformation) Wenn hier kein Abstand zum Anfang angegeben wird, so gelten die Werte für den gesamten Gitterträger.	T=300 I=9800
Wld_B2_Target	Sollwerte der Untergurt rechts-Schweißpunkte (siehe Schweißpunktinformation) Wenn hier kein Abstand zum Anfang angegeben wird, so gelten die Werte für den gesamten Gitterträger.	T=300 I=9800
Spacer	Informationen zu einem Abstandhalter (siehe Abstandhalterinformation)	H=20 D=95 X=3055.7 Y=282.1
CutBegin CutEnd WeldBegin WeldEnd BendBegin BendEnd PlaceBegin PlaceEnd	Es können Start- und/oder Endzeitpunkt diverser Produktionsereignisse angegeben werden. Definiert sind folgende Ereignisse: *Cut*: Schneiden auf Länge *Weld*: Verschweißen *Bend*: Biegen *Place*: Verlegen. Typischerweise gibt es pro Feedback-Block zu jedem dieser Ereignisse nur je einen Eintrag. D.h. bei einem *Steel*-Block beziehen sich die Angaben auf den gesamten *Steel*-Block.	2017-02-20T16:32:33+01:00
m2	Produzierte Fläche in Quadratmeter. Der Wert wird von der Maschine nach eigenen Kriterien ermittelt. Typischerweise wird die umhüllende xy-Box des produzierten Teils herangezogen.	17.32
RefItem_XX	Referenziertes Item. Hier ist **XX** nur ein Platzhalter und muss effektiv durch einen Tabellennamen ersetzt werden. Konkret gibt es also Typen wie RefItem_**Bar**, RefItem_**Segment**, usw. Der anzugebende Wert ist die GlobalID des referenzierten Items. Typischerweise werden solche Einträge bei PTS-Meldungen verwendet, die nehmen dem eigentlich betroffenen Item noch weitere „verursachende“ Items angeben. Ein Beispiel wäre eine Biegung, die nicht ausgeführt werden, kann, weil ein anderes Eisen zu nahe liegt. Dieses andere Eisen wäre dann ein Problemverursachendes Item, das das mittels RefItem_Bar angegeben kann.	12345ABC
HltBarTIntvl	Intervall auf Eisen (spezifiziert durch theoretische Längenpositionen), die einen	1320;1420

	Abschnitt angeben, der besonders hervorgehoben (highlighted) werden soll. Das wird typischerweise in PTS-Feedbacks verwendet, um beispielsweise eine Biegung hervorzuheben, bei der ein Problem auftritt. Da es sich um Positionswerte auf einem Eisen handelt, ist es für Bar-Items gedacht, oder für Items, die unterhalb der Bar-Ebene liegen. Die Reihenfolge, in der die beiden Intervallgrenzen angegeben werden, darf aufsteigend oder absteigend sein; der Anzeige-Client sollte eine Darstellung wählen, in der erkennbar ist, welches die erste und welches die zweite Intervallgrenze ist.	

3.14.4.1 Drahtinformation

Beinhaltet eine String, der sich aus folgenden Feldern zusammensetzt:

Feld	**Beschreibung**	**Beispiel**
D	Durchmesser [mm]	12
Qlty	Stahlqualität	BS300
Mtl	Coil-Info@Charge	83@C25408
Art	Artikel-Kode des Roh-Materials	GD12345
Len	Reale Länge [mm]	5730
Kg	Reales Gewicht [kg]	2.756

Beispiel: D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=7.756 Art=GD12345
Beachte: diese Information beschreibt das verbrauchte material (Roh-Material).

3.14.4.2 Schweißpunktinformation

Beinhaltet einen String, der sich aus folgenden Feldern zusammensetzt:

Feld	**Beschreibung**	**Beispiel**
X	Distanz X zum Anfang des geschweißten Teils (z.B. Matte , Gitterträger) [mm]	322
Y	Distanz Y zum Anfang des geschweißten Teils (z.B. Matte , Gitterträger) [mm]	257
I	Schweißstrom [mA]	8185
P	Schweißdruck [mBar]	9074
T	Schweißzeit [ms]	231
WH	ID des verwendeten schweißkopfes	1

Beispiel: X=0 I=9855.5 P=8861 T=221

3.14.4.3 Abstandhalterinformation

Beinhaltet einen String, der sich aus folgenden Feldern zusammensetzt:

Feld	**Beschreibung**	**Beispiel**
H	Höhe [mm]	25
D	Durchmesser [mm]	95
X	X-Position Bei „Slab“-Einträgen bezogen auf den Plattenanfang.	4234.9

Y	Y-Position Bei „Slab“-Einträgen bezogen auf den Plattenanfang.	122.7

Beispiel: H=25 D=95 X=4234.9 Y=122.7

Es können auch noch weitere interne Felder definiert werden. Diese sollten aber den Präfix „I_“ haben.

Die Reihenfolge der einzelnen Felder ist nicht bindend und nicht jedes Feld muss vorhanden sein.

Auch die Reihenfolge der verschiedenen Schweißpunkte bzw. Drähte untereinander ist nicht bindend und jeder Eintrag ist fakultativ.

3.14.5 Beispiele für PTS-Meldungen

Die genaue Liste der möglichen *PTS*-Meldungen wird vom Maschinenhersteller definiert. Die folgende Liste ist daher nur als Beispiel zu verstehen:

Beispiele für ItemType="Segment":

- Maximum segment length exceeded. [error]
- Segment too short. [error]
- Segment as L0 too short. [error]
- Bending too close as L0 too short. [error]
- Invalid bending angle. [error]

Beispiele für ItemType="Bar":

- Invalid bar diameter. [error]
- Bar with too few Welding points. [error]
- Bar too close to next bar. [error]
- Bar too close to next bar – corrected. [warning]
- Multiple bars aligned on same y; production will be slow. [hint]

Beispiele für ItemType="Steel":

- Invalid mesh size (too large). [error]
- Invalid mesh size (too small). [error]
- Mesh not transportable. [error]
- Mesh not transportable – corrected. [warning]

Da die Meldungen am CAD-Bildschirm eventuell nicht immer eindeutig visuell zuordenbar sind, sollte der *ItemType* aus dem aus dem *Description*-Text erkennbar sein.

Das folgende Beispiel zeigt das PXML-Dokument einer *PTS*-Rückmeldung:

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
  <DocInfo GlobalID="7C7E1FC5-0A46-48c5-A3B2-249D75B70BCF">
    <MajorVersion>1</MajorVersion>
    <MinorVersion>3</MinorVersion>
    <Comment>MSystem PTS 3.77.12.123, List 1, Params 2010-07-12</Comment>
  </DocInfo>
  <Feedback ItemType="Bar" GlobalID="12345">
    <MessageType>error</MessageType>
    <Code>MaxBarLen</Code>
    <Description Culture="en" Text="Maximum bar length exceeded."/>
    <Description Culture="de" Text="Max. Eisenlänge überschritten."/>
  </Feedback>
  <Feedback ItemType="Bar" GlobalID="2057">
```



```
    <MessageType>warning</MessageType>
    <Code>MinBarLen</Code>
    <Machine>BGM</Machine>
    <Description Culture="en" Text="Bar too short."/>
    <Description Culture="de" Text="Eisen zu kurz."/>
  </Feedback>
  <Feedback ItemType="Segment" GlobalID="5523">
    <MessageType>error</MessageType>
    <Code>DistCBar</Code>
    <Machine>BGM</Machine>
    <Description Culture="en" Text="Bending too near to crossing bar."/>
    <Description Culture="de" Text="Biegung zu nahe an querendem Eisen."/>
    <FbVal T="HltBarTIntvl" V="1320;1420"/>
  </Feedback>
</PXML_Document>
```

In obigem Beispiel sieht man 2 *PTS-Feedback*-Meldungen: eine Fehlermeldung für das Eisen "12345", und ein Warnung für das Eisen "2057". In diesem Beispiel sei "MinBarLen" ein Warnungs-Code, der Eisen betrifft, die von der Maschine automatisch verlängert werden; das Eisen kann also in automatisch korrigierte Form produziert werden.

Besonders hervorzuheben ist der *Comment*-Eintrag im *DocInfo*-Teil: hier sollte eine Identifikation des *PTS*-Servers stehen, aus der auch hervorgeht, wann seine Parameter zuletzt mit den Maschinen abgeglichen wurden. Idealerweise sollte der *PTS*-Client diese Information für den Benutzer sichtbar machen.

3.14.6 Beispiele für Machine-Return-Meldungen

Das folgende Beispiel zeigt das PXML-Dokument einer *Machine-Return*-Meldung, in der die Produktion zweier Eisen gemeldet wird.

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
  <DocInfo GlobalID="7C7E1FC5-0A46-48c5-A3B2-249D75B70BCF">
    <MajorVersion>1</MajorVersion>
    <MinorVersion>3</MinorVersion>
  </DocInfo>
  <Feedback ItemType="Bar" GlobalID="2057">
    <PieceCount>3</PieceCount>
    <MaterialType>16A</MaterialType>
    <MaterialBatch>12345@AR177228C</MaterialBatch>
    <ProdDate>2010-07-30T09:06:03+02:00</ProdDate>
  </Feedback>
  <Feedback ItemType="Bar" GlobalID="2058">
    <PieceCount>1</PieceCount>
    <MaterialType>8</MaterialType>
    <MaterialBatch>12345@AR177228C</MaterialBatch>
    <ProdDate>2010-07-30T09:06:05+02:00</ProdDate>
  </Feedback>
</PXML_Document>
```

Das folgende Beispiel zeigt das PXML-Dokument einer *Machine-Return*-Meldung, in der die Produktion von 3 Stück eines Elementes gemeldet wird; hierbei wird auch angegeben, welches Material verwendet wurde, und wie viele Kg benötigt wurden.

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
  <DocInfo GlobalID="7C7E1FC5-0A46-48c5-A3B2-249D75B70BCF">
    <MajorVersion>1</MajorVersion>
    <MinorVersion>3</MinorVersion>
  </DocInfo>
  <Feedback ItemType="Slab" GlobalID="12007">
    <PieceCount>3</PieceCount>
```



```
    <ProdDate>2010-07-30T09:06:03+02:00</ProdDate>
  </Feedback>
  <Feedback ItemType="Slab" GlobalID="12007">
    <MaterialType>10</MaterialType>
    <MaterialBatch>12345@AR177228C</MaterialBatch>
    <MaterialWeight>123.7</MaterialWeight>
    <ProdDate>2010-07-30T09:06:05+02:00</ProdDate>
  </Feedback>
  <Feedback ItemType="Slab" GlobalID="12007">
    <MaterialType>12</MaterialType>
    <MaterialBatch>12345@AR177228C</MaterialBatch>
    <MaterialWeight>162.4</MaterialWeight>
    <ProdDate>2010-07-30T09:06:05+02:00</ProdDate>
  </Feedback>
  <Feedback ItemType="Slab" GlobalID="12007">
    <MaterialType>KTS810</MaterialType>
    <MaterialWeight>98.7</MaterialWeight>
    <ProdDate>2010-07-30T09:06:05+02:00</ProdDate>
  </Feedback>
</PXML_Document>
```

Die Feedback-Blöcke beziehen sich hier alle auf den selben *Slab*-Eintrag. Der erste Feedback-Block gibt an, dass 3 Stück produziert wurden, die anderen melden Materialverbrauch, der für diesen *Slab*-Eintrag aufgewendet wurde.

Beachte: den Applikationen steht es frei, die Rückmeldungen auf grobe oder feine Hierarchiestufen zu beziehen. So ist es beispielsweise möglich, die Rückmeldung nur für den gesamten *Order*-Eintrag zu machen, also lediglich zurückzumelden, dass der *Order*-Eintrag abgearbeitet wurde, und eventuell wie viel und welches Material hierfür verwendet wurde. Andere Applikationen könnten dagegen Rückmeldungen auf *Bar*-Ebene machen. Wenn man aber verschiedene Hierarchiestufen zurückmeldet, so ist zu beachten, dass **summierbare Größen** (wie das Gewicht) sind immer **akkumulierbar** sein müssen: wenn also beispielsweise *Bar*-Feedbacks mit Gewicht gemeldet werden, und *Slab*-Feedbacks mit Gewicht gemeldet werden, so darf das gemeldete *Slab*-Gewicht nur jenes Zusatzgewicht beschreiben, das nicht in *Bar*-Feedbacks enthalten ist.

3.14.7 Beispiele für FbVal Einträge

3.14.7.1 Beispiel Bar

```
<Feedback ItemType="Bar" GlobalID="12345">
  <ProdDate>2010-07-30T09:06:05+02:00</ProdDate>
  <Machine>MSR1:2</Machine>
  <FbVal T="OrdNo" V="AA00048386"/>
  <FbVal T="ElemNo" V="5522A"/>
  <FbVal T="Size" V="5740.4"/>
  <FbVal T="ProdTy" V="4"/>
  <FbVal T="NomTy" V="M10"/>
  <FbVal T="Wr" V="D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=2.756"/>
</Feedback>
```

3.14.7.2 Beispiel Steel

```
<Feedback ItemType="Steel" GlobalID="12345">
  <ProdDate>2010-07-30T09:06:05+02:00</ProdDate>
  <Machine>MSystem:1</Machine>
  <FbVal T="OrdNo" V="AA00048386"/>
  <FbVal T="ElemNo" V="5522A"/>
  <FbVal T="Size" V="5740x1320"/>
  <FbVal T="ProdTy" V="4"/>
  <FbVal T="NomTy" V="M10"/>
```



```
  <FbVal T="Wr" V="D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=7.756"/>
  <FbVal T="Wr" V="D=16 Qlty=BS300 Mtl=83@C25408 Len=810 Kg=9.003"/>
</Feedback>
```

3.14.7.3 Beispiel Slab

```
<Feedback ItemType="Slab" GlobalID="12345">
  <ProdDate>2010-07-30T09:06:05+02:00</ProdDate>
  <Machine>MeshSpacer:1</Machine>
  <FbVal T="OrdNo" V="AA00048386"/>
  <FbVal T="ElemNo" V="5522A"/>
  <FbVal T="Area" V="5.5"/>
  <FbVal T="Spacer" V="H=25 D=95 X=122.3 Y=122.7"/>
  <FbVal T="Spacer" V="H=25 D=95 X=122.3 Y=650.6"/>
  <FbVal T="Spacer" V="H=25 D=95 X=122.3 Y=1333.2"/>
  <FbVal T="Spacer" V="H=25 D=95 X=1508.7 Y=122.7"/>
  <FbVal T="Spacer" V="H=25 D=95 X=1508.7 Y=650.6"/>
  <FbVal T="Spacer" V="H=25 D=95 X=1508.7 Y=1333.2"/>
  <FbVal T="Spacer" V="H=25 D=95 X=3296.8 Y=122.7"/>
  <FbVal T="Spacer" V="H=25 D=95 X=3296.8 Y=650.6"/>
  <FbVal T="Spacer" V="H=25 D=95 X=3445.7 Y=1234.8"/>
</Feedback>
```

3.14.7.4 Beispiel Girder

```
<Feedback ItemType="Girder" GlobalID="12345">
  <ProdDate>2010-07-30T09:06:05+02:00</ProdDate>
  <Machine>Versa1:2</Machine>
  <FbVal T="OrdNo" V="AA00048386"/>
  <FbVal T="ElemNo" V="5522A"/>
  <FbVal T="Size" V="800"/>
  <FbVal T="ProdTy" V="KT82012"/>
  <FbVal T="NomTy" V="KT82011"/>
  <FbVal T="Wr_Tp" V="D=12 Qlty=BS300 Mtl=99@C25408 Len=810 Kg=2.756"/>
  <FbVal T="Wr_B1" V="D=6 Qlty=BS300 Mtl=81@C25422 Len=850 Kg=0.689"/>
  <FbVal T="Wr_B2" V="D=6 Qlty=BS300 Mtl=80@C250 Len=850 Kg=0.689"/>
  <FbVal T="Wr_D1" V="D=5 Qlty=BS300 Mtl=83@C2555 Len=1212.7 Kg=0.95"/>
  <FbVal T="Wr_D2" V="D=5 Qlty=BS300 Mtl=94@C25476 Len=1212.7 Kg=0.95"/>
  <FbVal T="Wld_B1" V="X=0 I=9876.5 P=8862 T=230"/>
  <FbVal T="Wld_B2" V="X=0 I=9855.5 P=8861 T=221"/>
  <FbVal T="Wld_Tp" V="X=100 I=9855.5 P=8861 T=340"/>
  <FbVal T="Wld_B1" V="X=200 I=9876.5 P=8862 T=433"/>
  <FbVal T="Wld_B2" V="X=200 I=9855.5 P=8861 T=255"/>
  <FbVal T="Wld_Tp" V="X=295 I=9855.5 P=8861 T=340"/>
  <FbVal T="Wld_B1" V="X=410 I=9816.5 P=6862 T=221"/>
  <FbVal T="Wld_B2" V="X=410 I=9855.1 P=8861.6 T=533"/>
  <FbVal T="Wld_Tp" V="X=500 I=8153.5 P=9871 T=340"/>
  <FbVal T="Wld_B1" V="X=610 I=9816.5 P=8862 T=113.4"/>
  <FbVal T="Wld_B2" V="X=610 I=9855.1 P=7861.6 T=60"/>
  <FbVal T="Wld_Tp" V="X=700 I=9458.3 P=9848.4 T=340"/>
  <FbVal T="Wld_B1" V="X=790 I=7200.5 P=8862 T=222"/>
  <FbVal T="Wld_B2" V="X=791 I=9855.1 P=8761.6 T=221"/>
  <FbVal T="Wld_Tp_Target" V="T=340 I=10200 P=8750"/>
  <FbVal T="Wld_B1_Target" V="T=300 I=9800 P=6000"/>
  <FbVal T="Wld_B2_Target" V="T=300 I=9800 P=7000"/>
</Feedback>
```


3.14.8 Kommunikations-Arten des PTS

Für die Kommunikation zwischen *PTS*-Client und *PTS*-Server gibt es viele Möglichkeiten. Jedes *PTS*-Server-System sollte aber mindestens die im folgenden angeführten Möglichkeiten anbieten.

3.14.8.1 Kommunikation auf File-Basis

Client und Server kommunizieren über zwei Verzeichnisse, ein **Request**-Verzeichnis und ein **Feedback**-Verzeichnis. Der Server beobachtet das *Request*-Verzeichnis; sobald dort eine Datei liegt, wird die Datei gelesen und gelöscht, und im *Feedback*-Verzeichnis wird eine Antwort-Datei bereitgestellt, die den selben Namen hat, wie die Anfrage-Datei.

Der Server bietet die Möglichkeit beliebig viele *Request/Feedback*-Verzeichnis-Paare zu definieren, so dass es für jeden Client ein eigenes Verzeichnis-Paar gibt.

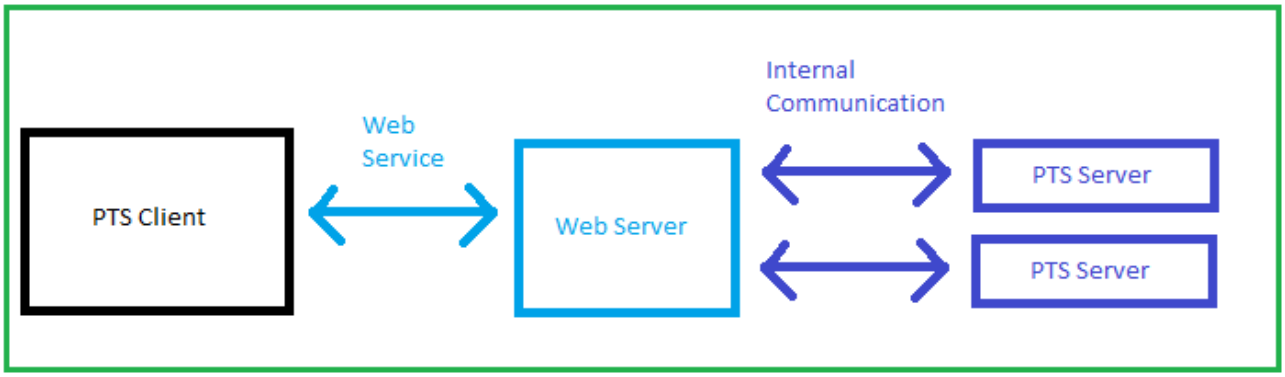
Jedes *Request/Feedback*-Verzeichnis-Paar entspricht am *PTS*-Server auch einem Parameter-Block (Produktionsliste, Produktionsmodus[70], usw.); es kann daher vorkommen, dass ein *PTS*-Client unterschiedliche *Request/Feedback*-Verzeichnis-Paare anspricht.

3.14.8.2 Kommunikation über Webservices

(Hier wird das Prinzip der PTS-kommunikation über Web-Services erklärt. Die vollständige formale Definition der PXML Web API findet man auf www.pxml.eu).

Client und Server kommunizieren über Webservices, die vom Server zur Verfügung gestellt werden.

Man muss hier zwischen **Web-Server** und **PTS-Server** unterscheiden: der Web-Server dient als **Gateway**, der die Webservices nach außen zur Verfügung stellt und die Abfragen dann intern an die dahinterliegenden eigentlichen PTS-Server weiterleitet.



Login

Um die Kommunikation zu starten, fordert der Client ein **Authentifizierungstoken** an, und zwar über einen **Login-Service**:

POST https://*mydomain.org*/Authentication/login

Authorization: Basic *dXNlclhZOnBhc3N3b3JkQUJD*

Hierbei ist "*mydomain.org*" die Adresse des Dienstes und „*dXNlclhZOnBhc3N3b3JkQUJD*" ist eine in Base64 codierte Sequenz *Benutzername:Passwort* (in obigem Beispiel wurde als Benutzername „userXY" verwendet, und als Passwort „passwordABC").

Das Authentifizierungstoken wird in dem Header der http-Response zurück geliefert. Das erhaltene Authentifizierungstoken muss für alle weiteren APIs in dem Header der Abfragen eingefügt werden (Token: {token}).

[70] Ein Beispiel einer *Produktionsmodus*-Variante ist folgendes: manche Client-Systeme wollen bei der PTS-Anfrage, dass die Maschinen als zu 100% verfügbar betrachtet werden (diese Anforderung ist für CAD-PST-Clients typisch); andere PTS-Clients (z.B. Taktplatzansichten von Leitrechner) wollen dagegen, dass die Maschinen mit ihrem aktuellen Betriebszustand berücksichtigt werden, dass also beispielweise ein deaktivierter Biegekopf berücksichtigt wird.

PTS-Ressource am Server anlegen (= PTS-Anfrage senden):

Der Client setzt eine PTS-Anfrage am Server als neue Ressource ab, auf die der Clent dann in der Folge Bezug nehmen kann:

POST https://*mydomain.org*/PTS

Als Body-Parameter übergibt man das **PXML-Dokument** als Byte-Array und einen **PtsServerPath**. Der PtsServerPath ist ein allgemeiner String, der den PTS-Server identifiziert. Das ist notwendig, weil über einen einzigen PTS-Web-Dienst mehrere PTS-Server angesprochen werden können – jeder dieser PTS-Server stellt eine eigene Maschine oder Maschinenkonfiguration dar.

Diese Parameter werden im Body der Abfrage in JSON oder XML formatiert.

Beispiel JSON:

```
{
    "PtsServerPath": "MSytem/List1",
    "PxmlDocument": "QEA="
}
```

(Hier ist „QEA=“ ein Beispiel eines über Base64 codierten Byte-arrays, das dem PXML-Dokument entspricht).

Der Client bekommt eine **TestID** zurück.

Anfrage zum Status der PTS-Ressource:

Über einen Abfrage-Service ist es möglich, den Stand der PTS-Verarbeitung einzusehen:

GET https://*mydomain.org*/ PTS/{*TestID*}/Status

Diese Abfrage liefert den Status der PTS-Test-Ressource als String zurück. Folgende Werte sind möglich:

- **WaitingForFeedbackFromPtsServer**: Das PXML-Dokument wurde an den PTS Server gesendet und es wird auf das Feedback des PTS-Servers gewartet.
- **RequestNotSentToPtsServer**: Es ist nicht gelungen, das PXML-Dokument zum angegebenen PTS-Server weiterzuleiten.
- **FeedbackAvailable**: Der PTS-Test wurde durchgeführt und das Ergebnis steht bereit.
- **RequestToPtsServerTimedOut**: Der PTS-Server hat nicht in der erwarteten Zeit geantwortet. Der Vorgang wurde abgebrochen.

Abrufen des Feedbacks:

Sobald das Feedback auf dem Web-Server verfügbar ist (Status *FeedbackAvailable*), kann es eingesehen werden:

GET https://*mydomain.org*/ PTS/{*TestID*}/Feedback

Diese Abfrage liefert den die PXML-Feedback-Datei als Byte-Array.

Löschen einer PTS- Ressource:

Wenn die PTS-Ressource nicht mehr benötigt wird, sollte sie gelöscht werden:

DELETE https://*mydomain.org*/ PTS/{*TestID*}

Eventuell kann der Server nicht gelöschte PTS-Ressourcen auch nach einiger Zeit automatisch löschen.

3.14.9 Kommunikations-Arten des Machine Return

3.14.9.1 Kommunikation auf File-Basis

Die Maschine schreibt fortlaufend *PXML*-Dateien, die übergeordneten Leitsystem gelesen und optional gelöscht werden.

Um den lesenden System die Auswertung zu erleichtern, sollten die Dateinamen die Sequenz der geschriebenen Dateien wiedergeben (entweder durch einen Timestamp im Dateinamen oder durch eine fortlaufende ID im Dateinamen).

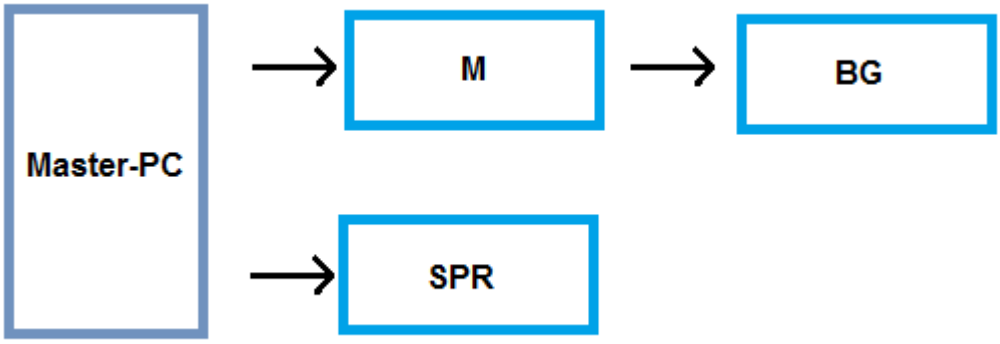
Der Maschine steht es frei, für jede *Feedback*-Meldung eine eigene Datei zu generieren, oder mehrere Feedback-Meldungen in eine Datei zu packen. Eine Datei, die einmal geschrieben wurde darf aber nachträglich nicht mehr verändert werden. D.h. die Maschine darf eine bestehende Datei nicht neuerlich öffnen, etwa um weitere *Feedback*-Blöcke anzuhängen. In diesem Punkt unterscheidet sich dieses Konzept ganz klar von den Regeln der Unitechnik-Report-Files.

3.14.9.2 Kommunikation über Web-Services

Siehe www.pxml.eu.

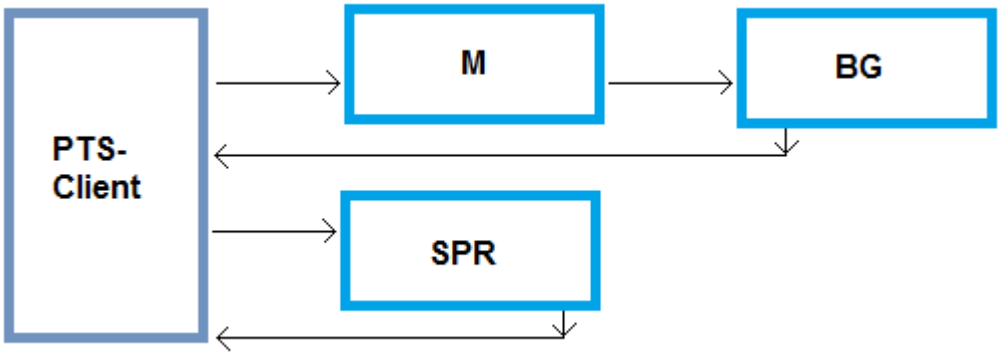
3.14.10 Parallel- und Reihenschaltung von PTS-Server

Eine komplexe Produktionsanlage kann unter Umständen mehrere PTS-Server haben. So könnte es beispielsweise eine Mattenschweißanlage M mit nachfolgender Mattenbiegeanlage BG geben, und parallel dazu einen Schalungsroboter SPR:



Die Maschine M und die Maschine BG sind hintereinandergeschaltet, in dem Sinne, dass BG von M mit Daten versorgt wird. Das Leitsystem ("Master-PC") schickt Daten parallel zu M und zu SPR.

Die 3 Maschinen werden durch 2 parallele PTS-Server abgebildet:



Ein PTS-Client sollte in diesem Sinne in der Lage sein, mehrere parallele PTS-Server abzufragen.

Die Kombination der Maschinen M/BG stellt dagegen nach außen nur ein PTS-System dar: die Weitergabe der PTS-Feedback-Datei von M nach BG ist von außen nicht erkennbar. Diese Verkettung ist nötig, da M die Daten eventuell geringfügig korrigiert, und BG mit den korrigierten Daten weiterarbeiten muss. Demgegenüber arbeitet SPR ganz unabhängig von M.

3.14.11 Filtern, klassifizieren und sortieren von PTS-Meldungen

Der PTS-Server nimmt nur eine grobe Klassifizierung der Meldungen vor, indem er sie in Fehler, Warnungen, Hinweise und Informationen einteilt (Siehe Abschnitt 3.14.3.2). Da der Server im Prinzip mehrere Clients bedienen kann, die unter Umständen unterschiedliche Aufgaben und Konzepte verfolgen, ist es nicht Aufgabe des Servers, PTS-Meldungen zu filtern, zu sortieren oder genauer zu klassifizieren. Der PTS-Server meldet alles aus Sicht der Maschine: er sagt, was produzierbar ist, was nicht, und was in eingeschränkter Form – er bewertet nicht, welche dieser Meldungen wichtig oder unwichtig sind.

Es ist Aufgabe des Clients, Möglichkeiten vorzusehen, PTS-Meldungen geeignet darzustellen. Dazu benötigt der Client Mechanismen, um PTS-Meldungen zu filtern und zu sortieren. Idealerweise gibt es auch die Möglichkeit, bestimmte Meldungen besonders hervorzuheben.

4 Vorschläge für zukünftige Erweiterungen

4.1 *ProdControl*-Tabelle für die Produktionssteuerung

Die CAD-Daten für die Produktion werden im Wesentlichen über die Order-Tabelle (samt Untertabellen) vermittelt. Diese Daten beschreiben die zu produzierenden Elemente, ohne aber Informationen über den Echtzeit-Produktionssteuerung zu geben.

Die *ProdControl*-Tabelle enthält dagegen Echtzeitinformationen für die Subsysteme und dient somit der direkten Produktionssteuerung. Eine Datei, die *ProdControl*-Einträge enthält ist vergleichbar mit dem UNICAM-Imagefile.

Typischerweise wird eine *ProdControl*-Datei vom Leitsystem geschrieben und von den Subsystemen gelesen. Da es sich um Beinahe-Echtzeit-Informationen handelt, werden solche Dateien immer wieder vom Leitsystem aktualisiert. Die Subsysteme müssen daher damit rechnen, dass die Datei temporär gesperrt ist; temporäre Zugriffsverweigerungen müssen somit toleriert werden (übrigens auch vom Leitsystem, denn auch der lesende Subsystem-Prozess kann die Date sperren).

Grundsätzlich kann eine *ProdControl*-Datei auch von mehreren Subsystemen gemeinsam genutzt werden. Flexibler ist man aber, wenn für jedes Subsystem eine individuelle *ProdControl*-Datei geschrieben wird.

Ein *ProdControl*-Eintrag beschreibt im Wesentlichen eine zu produzierende Fertigungseinheit, einschließlich spezifischer Produktionsanweisungen an die Subsysteme. Die Fertigungseinheit kann hierbei einer Produktionslinien-Taktplatz-Station zugeordnet werden.

4.1.1 Felder der *ProdControl*-Tabelle

- **StationNo:** Identifizierender Name der Taktplatz-Station (Stationsnummer oder Stationsname).
  Neben echten (physisch existenten) Umlaufstationen kann es auch virtuelle *Vorlauf*-Plätze geben. Für diese wird typischerweise keine Stationsnummer angegeben, oder die Stationsnummer "000".
  Man beachte, dass eine *StationNo* auch mehrmals in einer Datei vorkommen kann. Es gibt dann zu dieser Station mehrere Fertigungseinheiten. Typischerweise wird das aber nur für *Vorlauf*-Plätze verwendet; für *echte* Stationen hat man meistens nur einen *ProdControl*-Eintrag in der Datei.
- **ProdSequence:** Alphanumerischer Reihenfolge-Schlüssel, der die Produktionsreihenfolge beschreibt. Dort wo dieser Schlüssel nicht eindeutig ist (insbesondere auch dann, wenn dieses Feld in der gesamten Datei nicht gesetzt wurde), zählt die Reihenfolge der *ProdControl*-Einträge als Reihenfolge.
  Bei Umläufen mit strikt sequentieller Stations-Anordnung werden die Subsysteme häufig eine festgelegte Stationsreihenfolge verwenden, um die Produktionsreihenfolge zu bestimmen. Bei Umläufen mit Verzweigung ist die *ProSequence*-Angabe dagegen oft unentbehrlich.
- **PalNo:** Identifikation der physischen Produktionspalette. Diese Information hat bei Umlaufanlagen eine entscheidende Bedeutung für den Echtzeithandshake: Ein *ProdControl*-Eintrag wird vom Subsystem genau dann als aktuell angesehen, wenn die *PalNo* aus der Datei mit der *PalNo* übereinstimmt, die über den Echtzeit-Feldbus gemeldet wird. Erst diese Überprüfung verleiht der *ProdControl*-Datei Echtzeit-Qualität.
  Für Umlaufanlagen, die nicht über die Möglichkeit verfügen, mittels Feldbus eine *PalNo* zu liefern, muss auf diese exakte Überprüfung verzichtet werden. Bei diesen Anlagen sollte das Leitsystem aber sicherstellen, dass die *ProdControl*-Datei aktualisiert wird, bevor ein Palette dem Subsystem als '*bereit*' gemeldet wird. Das Subsystem muss dann aber damit

rechnen, dass diese Bedingung in Ausnahmefällen verletzt wird (z.B. bei Netzwerk-Störungen oder beim Leitrechner-Neustart).
Bei Vorlauf-Einträgen (keine *StationNo*) wird typischerweise auch keine *PalNo* angegeben.

- **PalType:** Typ der physischen Palette. Das ist z.B. relevant, wenn es Produktionspaletten mit und ohne feste Randschalung gibt, oder wenn Paletten mit unterschiedlichen Abmessungen im Umlauf sind.
- **ProductType:** Hier kann ein Produkt-Typ angegeben werden, z.B. unter Verwendung der Kürzel, die im Abschnitt 3.5.2 angegeben wurde (z.B. 00, 01, DW, 09, TW, ...).
Dieses Feld ist im Wesentlichen aus Kompatibilität zum UNICAM-Imagefile vorgesehen. Abgesehen davon besteht aber kaum Veranlassung das Feld zu verwenden, da der Produkt-Typ bereits in der CAD-Datei festgelegt wird.
- **X/Y:** Über die X- und Y-Werte kann ein optionaler Belegungsoffset angegeben werden, der den CAD-Daten aufzurechnen ist.
- **DataKey:** Ein eindeutiger Schlüssel, der die Fertigungseinheit identifiziert. Hier kann beispielsweise die Paletten-Daten-ID des Leitsystems stehen, eventuell noch ergänzt durch weitere Informationen. Entspricht im UNICAM-Imagefile dem zusammengesetzten Schlüssel aus 'Fertigungseinheit' und 'Teil'.
Der *DataKey*-Wert sollte vom Subsystem auch verwendet werden, um wiederholtes Importieren der selben Daten zu vermeiden: wenn die Daten zu einem *DataKey*-Wert bereits in der subsysteminternen Datenbank vorhanden sind, müssen sie nicht mehr importiert werden.
- **CadData:** Dateiname der CAD-Datei, die die Produktionsdaten enthält. Es kann ein einfacher Dateiname in einem als bekannt vorausgesetzten Verzeichnis genannt werden (eventuell mit relativer Pfadangabe), oder es kann ein allgemeiner UNC-Pfad angegeben werden.
- **PieceCount:** Optionaler Soll-Stückzahl-Multiplikator. Wird typischerweise nur bei Stapel-Produktionen verwendet (nicht bei Paletten-Umlauf-Produktionen).
- **NoData:** Ein Wert von *true* sagt hier aus, dass die Palette ohne Produktions-Aktionen durch das Subsystem durchfahren soll. Typischerweise werden für eine solche Palette auch keine Produktionsdaten angegeben (Keine Angabe von *DataKey* und *CadData*). Das alleinige Fehlen von Produktionsdaten darf im Allgemeinen aber noch nicht als implizite *NoData*-Anweisung verstanden werden, da die Datenzuweisung gelegentlich erst verspätet erfolgt.
Man beachte, dass eine *NoData*-Anweisung lediglich darauf hinweist, dass keine Produktionsdaten-bezogenen Operationen durchzuführen sind. Dem Subsystem steht es aber dennoch frei, andere Operationen an der betreffenden Palette auszuführen (Schalungen entnehmen, Palette reinigen, Überschuss auf Palette entsorgen).
- **Wait:** Ein Wert von *true* gibt an, dass das Subsystem mit der Bearbeitung warten soll. Die Warte-Anweisung bezieht sich auf den *Haupt*-Vorgang der Verarbeitung. Gewisse vorbereitende Arbeiten können dennoch ausgeführt werden.
Die Funktion kann z.B. beim Betonverteiler verwendet werden, um das Betonieren der Palette zu unterbinden, bis eine Betonier-Freigabe erfolgt. Schon vorher kann der Betonverteiler aber beim Mischer eine Betonbestellung auslösen.
- **ExtraSize:** Ein Wert von *true* gibt an, dass die Palette in Übergröße belegt werden darf, weil das bei ihrer weiteren Bearbeitung berücksichtigt werden wird. Typisches Beispiel: eine Palette darf vom Schalungsroboter außerhalb ihrer normalen Begrenzungen belegt werden, wenn die Palette dann nicht in die Trockenkammer fährt.

4.1.2 *ProdDirective*-Untertabelle

Die *ProdDirective*-Tabelle ist eine Untertabelle von *ProdControl* und enthält Anweisungen, die als Liste formuliert werden müssen.

Ein *ProdDirective*-Eintrag hat folgende Felder:

- **Code:** Der Anweisungscode.
- **Val:** Wert, der Abhängig vom Anweisungscode zu interpretieren ist.

Für den *Code* gibt es Standard-Definitionen und anwendungsspezifische Definitionen. Letztere sollten einen Code verwenden, der mit "I_P_" beginnt, um potentielle Konflikte mit Erweiterungen des Standards zu vermeiden.

Der Standard definiert derzeit nur *ProdDirectives* zur Filterung. Siehe unten.

4.1.2.1 Filter-ProdDirectives

Das *ProdDirective.Code*-Feld enthält einen Text der Form

filter:<Table>.<Field>

Das *ProdDirectives.Val*-Feld enthält einen Textfilter-Ausdruck in Form einer *Regular Expression*.

Beispiel:

```
  <ProdControl>
    <ProdDirective>
      <Code>filter:Slab.PartType</Code>
      <Val>2</val>
    </ProdDirective>
    <ProdDirective>
      <Code>filter:Steel.MeshType</Code>
      <Val>4</val>
    </ProdDirective>
  </ProdControl>
</PXML_Document>
```

Diese Filterangaben würde dazu führen, dass nur die Daten der zweiten Wandhälfte verarbeitet werden, und bei der Bewehrung nur die Deckelmatte (Mattentyp 4) aktuell einzulegen ist[71].

Mit Hilfe solcher Filter-Direktiven lassen sich auch mehrstufige Produktionsprozesse an einer Palette gut steuern. So z.B. wenn in einem ersten Bearbeitungsschritt nur die unteren Lagen beachtet werden, und in einem zweiten Bearbeitungsschritt die oberen Lagen[72].

Prinzipiell kann ein Leitsystem die Filterung auch direkt auf die CAD-Datei anwenden, ohne die Filter-Direktiven der *ProdControl*-Datei zu verwenden. Das geht aber nicht immer, denn das Subsystem benötigt manchmal nicht nur die aktuell zu produzierenden Daten, sondern auch weitere damit im Zusammenhang stehenden Informationen (So kann ein Schalungsroboter beim Schalen einer Wandhälfte beispielsweise auch Informationen der zweiten Wandhälfte benötigen).

[71] Wenn man die ausführlichen Möglichkeiten der *Regula Expression* Textfilter ausschöpft, kann man die Filter noch präziser formulieren. Die erste Wandhälfte würde durch den TextFilter ^\s*(02|2)\s*$ sehr exakt beschrieben.

[72] Im UNICAM-Imagefile wird das meist über die inoffizielle Erweiterung "pass2" realisiert.

4.1.3 Beispiel einer *ProdControl* Datei


```xml
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
  <DocInfo GlobalID="7C7E1FC5-0A46-48c5-A3B2-249D75B70BCF">
    <MajorVersion>1</MajorVersion>
    <MinorVersion>3</MinorVersion>
  </DocInfo>
  <ProdControl>
    <StationNo>12</StationNo>
    <ProdSequence>A123567</ProdSequence>
    <PalNo>43</PalNo>
    <PalType>2</PalType>
    <ProductType>DW</ProductType>
    <X>0</X>
    <Y>100</Y>
    <DataKey>00000024542-41477[1.8.1]</DataKey>
    <CadData>00024542.pxml</CadData>
    <PieceCount>1</PieceCount>
    <NoData>False</NoData>
    <Wait>False</Wait>
    <ExtraSize>False</ExtraSize>
    <ProdDirective>
      <Code>filter:Slab.PartType</Code>
      <Val>^\s*(02|2)\s*$</val>
    </ProdDirective>
    <ProdDirective>
      <Code>filter:Steel.MeshType</Code>
      <Val>^\s*4\s*$</val>
    </ProdDirective>
  </ProdControl>
  <ProdControl>
    <StationNo>10</StationNo>
    <PalNo>44</PalNo>
    <NoData>True</NoData>
  </ProdControl>
  <ProdControl>
    <ProdSequence>A123570</ProdSequence>
    <DataKey>00000024570-41477[1.8.1]</DataKey>
    <CadData>00024570.pxml</CadData>
  </ProdControl>
  <ProdControl>
    <ProdSequence>A123571</ProdSequence>
    <DataKey>00000024571-41477[1.8.1]</DataKey>
    <CadData>00024571.pxml</CadData>
  </ProdControl>
  <ProdControl>
    <ProdSequence>A123572</ProdSequence>
    <DataKey>00000024572-41477[1.8.1]</DataKey>
    <CadData>00024572.pxml</CadData>
  </ProdControl>
</PXML_Document>
```


4.2 Kommunikation mit Mischer-Anlagen

4.2.1 Concrete Order (Concrete Distributer → Batching Plant)

Each concrete order is sent to the batching plat by a small XML file:

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
  <DocInfo>
    <MajorVersion>1</MajorVersion>
    <MinorVersion>3</MinorVersion>
  </DocInfo>
  <Order>
    <Product>
      <ElementInfo Type="ConcreteLot">
        <ElemInfoVal Type="Recipe" V="B25"/>
        <!-- Volume in m³: -->
        <ElemInfoVal Type="Volume" V="1.234"/>
        <!-- Delivery position, e.g. ID of a flying bucket: -->
        <ElemInfoVal Type="Target" V="U2"/>
        <!-- Optional mixer-ID (for multiple mixer batching plants): -->
        <ElemInfoVal Type="Mixer" V="1"/>
      </ElementInfo>
    </Product>
  </Order>
</PXML_Document>
```

4.2.1.1 Communication by file transfer

The file transfer handshake is as follows:

a) The system issuing the concrete order (concrete distributor, MES system, ERP system) puts the order file into a ConcreteOrder folder (Shared Network Folder). The filename should be an ascending alphanumerical key, in order to guarantee a well-defined sequence. The filename should start with “a_”.

b) When the batching plant has read in the file, it renames the prefix of the file from “a_” to “b_“.

c) When the batching starts to process the order, it renames the prefix of the file from “b_” to “c_“. (Some batching plants may not be able to distinguish between the “b_” and the “c_” state. In such case, they would immediately signal the “c_” state after having read in the file).

d) When the batching plant delivers the concrete to the transport unit (flying bucket), the filename is changed again, this time by changing the prefix to “t_“. This renaming process is again done by the batching plant.

e) When the target system (e.g. the concrete distributor) receives the concrete batch, it deletes the “t_”-file. If the target system isn’t able to perform such a file manipulation task, file deletion should already be done by the batching plant, when delivering the concrete to the transport unit (i.e. in such a case instead of renaming the file to “t_”, the file should simply be deleted).

f) If the batching plant doesn’t accept the concrete order (wrong file content), the file should be renamed the way to have a prefix “x_“. Such an erroneous file needs to be deleted by the system that issued the mixing order (e.g. the concrete distributor).

g) There are individual ConcreteOrder folders for each ordering system and each batching plant (if, for instance, there are 3 concrete distributors and two batching plants, there are in total 6 separated ConcreteOrder folders). It is recommended to have only one “a_” or “x_” file in each ConcreteOrder folder at any time; doing so, it is avoided that an error condition (“x_” file) leads to alterations of the required production sequence.

h) A single batching plant may have more than one mixers. In such a case the concrete order file may (optionally) specify the mixer to be used. Alternative options may be specified by using a syntax like "1|2".

4.2.1.2 Communication by web-service

To be defined

4.2.2 Production Feedback (Batching Plant → ERP-System)

For each mixture that is produced, the batching plant creates a "*Machine Return*" feedback with the following structure (see general description of *Machine Return* feedbacks for handshake details, section 3.14.9).

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
  <DocInfo>
    <MajorVersion>1</MajorVersion>
    <MinorVersion>3</MinorVersion>
  </DocInfo>
  <Feedback ItemType="ConcreteLot">
    <ProdDate>2010-07-30T09:06:05+02:00</ProdDate>
    <!--BatchingPlant and optional internal Mixer:-->
    <Machine>BachingPlantXY:2</Machine>
    <!--Internal reference number of the mixture:-->
    <FbVal T="ProdRef" V="AA00048386"/>
    <!--Recipe:-->
    <FbVal T="ProdTy" V="B25"/>
    <!--Volume in m³:-->
    <FbVal T="Size" V="1.234"/>
    <FbVal T="Target" V="U2"/>
    <!--Raw-Material-List:-->
    <!--The Code identifies the (material water, aggregate, cement, ...)-->
    <!--The Quantity is related to the given Unit (m3, L, Kg)-->
    <FbVal T="RawMaterial" V="C=123A5 Qty=0.23 Unit=m3 Storage=12A"/>
    <FbVal T="RawMaterial" V="C=Wtr Qty=123 Unit=L"/>
    <FbVal T="RawMaterial" V="C=CEE3 Qty=0.224 Unit=Kg Storage=15A"/>
  </Feedback>
</PXML_Document>
```

4.2.3 List of available raw material articles (ERP-System → Batching Plant)

In order to help assuring the concrete receipts created in the batching plant use only article numbers available in the ERP system, the ERP system may (optionally) provide a list of all available raw material article codes. The list is grouped in ConcreteComponent blocks, where each block is associated to a weighting unit of the batching plant.

The whole information is provided by one or several files of with the following structure[73]

```
<?xml version="1.0" encoding="utf-8"?>
<PXML_Document xmlns="http://progress-m.com/ProgressXML/Version1">
  <DocInfo>
    <MajorVersion>1</MajorVersion>
    <MinorVersion>3</MinorVersion>
  </DocInfo>
  <Feedback ItemType="ConcreteComponents:XY"><!--"XY"=ID of weighting unit -->
    <FbVal T="RawMaterial" V="C=123A5 Unit=m3"/>
    <FbVal T="RawMaterial" V="C=Wtr Unit=L"/>
    <FbVal T="RawMaterial" V="C=CEE3 Unit=Kg"/>
  </Feedback>
```

[73] It is left open to individual implementations if one file is provided which contains all ConcreteComponent blocks, or if the ConcreteComponents blocks are spread over different files.


```
  <Feedback ItemType="ConcreteComponents:AB"><!--"AB"=ID of weighting unit -->
    <FbVal T="RawMaterial" V="C=1755 Unit=m3"/>
    <FbVal T="RawMaterial" V="C=A1933 Unit=Kg"/>
  </Feedback>
</PXML_Document>
```

5 Versions-History

Version 1.1

Neue Felder im *Bar*-Abschnitt

- **Bin:** Fach

Version 1.2

Neue Felder im *Bar*-Abschnitt

- **Pos:** Position
- **Note:** Bemerkung
- **Machine:** Maschinenzuweisung
- **BendingDevice:** Biegevorrrichtung
- **NoAutoProd:** Ersetzt das *AutomaticProduction*-Feld (ist aber invertiert)

Neue Felder im *Girder*-Abschnitt

- **NoAutoProd:** Ersetzt das *AutomaticProduction*-Feld (ist aber invertiert)

Felder im *Slab*-Abschnitt wurden entfernt

- LateralFormworkType
- LongitudinalFormworkType
- PlasterThicknessBottom
- PlasterQualityBottom
- UnitWeightBottom
- PlasterVolumeBottom
- PlasterThicknessTop
- PlasterQualityTop
- UnitWeightTop
- PlasterVolumeTop

Neue *Outline*-Felder

- **Z**-Koordinate. Ersetzt *MountPartInstallHeight*
- **Height**. Ersetzt *MountPartThickness*.

Entfernte *Outline*-Felder

- MountPartThickness
- MountPartInstallPosX
- MountPartInstallPosY
- MountPartInstallHeight
- Area

Neuer *Outline*-Typ: *lot*
Dient zur Darstellung von Beton-Parzellen. Ersetzt auch die Typen *contout* und *cutout*.

Neues *Shape*-Feld: *Cutout*
Wenn gesetzt, ist das entsprechende Shape-Polygon eine Aussparung

Entfernen der Concrete-Layer-Elemente

Es gibt keine Concrete-Layer-Elemente mehr, da die Betonschicht-Eigenschaften in der *lot*-Outline enthalten ist. Beim Export nach UNICAM muss eine globale Betonschicht angegeben werden; hier wird die Information des ersten *lot*-Eintrags noch mal koiert.

Der *LayerType* (Kennzeichen der Schicht) entfällt und wird nicht mehr unterstützt.

Entfernen der *contour*- und *cutout*-Outline-Typen.

Diese Outline-Typen werden nicht mehr benötigt, da durch *lot*-Outline ersetzt.

Wenn es in UNICAM eine Gesamt-Kontur mit mehreren Betonschichten gibt, so wird dies in einzelne flächenkongruente *lot*-Elemente zerlegt.

Neu-Definition der Multi-Layer-Elemente.

Einführung des **MasterLayer**-Flags; Neudefinition des Multi-Layer-Konzepts in PXML

Default-Werte sind für jeden Datentyp einheitlich festgelegt

Alle individuellen Defaultwert-Festsetzungen sind damit außer Kraft gesetzt.

Version 1.2 – Later Additions

Neue Felder im *Steel*-Abschnitt (Added on 2007-08-14)

- **Steel.WeldingDensity:** Schweißdichte.
- **Steel.BorderStrength:** Stärke des Matten-Randes.

Neues Feld im Bar-Abschnitt (Added on 2007-09-25)

- **Bar.Ident:** Numerische ID für Systemweite Identifikation.

Neue Felderim Steel-Abschnitt (Added on 2007-10-03)

- **Steel.GenericInfo03…06:** Freie Informationsblöcke (bisher gab es nur 2 Info-Blöcke).

Neuer Steel-Typ (Added on 2007-10-13)

- **Steel.Type = "cage":** Korb für Korbfertigungsanlagen.

Alloc-Tabelle eingeführt (Added on 2007-11-17)

- **Tabellen "Alloc" und "Region":** Verlegungs-Muster für Bügel, Staffeleisen und Gitterträger.

SteelExt-Tabelle eingeführt (Added on 2007-12-05)

- **Tabelle "SteelExt":** Collection von applikationsspezifischen Zusatz-Daten zum Steel-Block.

Neues Feld im Girder-Abschnitt (Added on 2008-07-21)

- **Girder.Machine:** Text-Feld für Maschinen- und Listen-Angabe.

Neue Felderim Girder-Abschnitt (Added on 2008-11-28)

- **Girder.Period:** Gitterträger-Periode.
- **Girder.PeriodOffset:** Abstand der des ersten Perioden-Tiefpunktes vom Gitterträger-Anfang.

GirderExt-Tabelle eingeführt (Added on 2008-12-12)

- **Tabelle "GirderExt":** Generische Untertabelle zur GirderRow.

Neue Felderim Girder-Abschnitt (Added on 2009-02-18)

- **Girder.GirderType:** Bedeutung dieses Feldes genauer festgelegt. Schubträger-Typ definiert.
- **Girder.MountingTyp:** Einbau-Art für Gitterträger.

Neues Feld im Girder-Abschnitt (Added on 2009-03-10)

- **Girder.BottomFlangeDiameter:** Untergurt-Durchmesser in mm.

Neue Felderim Girder-Abschnitt (Added on 2009-03-19)

- **Girder.TopExcess, Girder.BottomExcess:** Überstand des Perioden-Eisens gegenüber Ober- und Untergurt.

Export nach UNICAM für GirderExtRows (Added on 2009-04-01)

- Der Export erfolgt über Gitterträgerschweißzeilen.

Neues Feld im Product-Abschnitt (Added on 2010-02-11)

- **Product.TurnWidth:** Angenommene Palettenbreite für Doppelwände.

Neues Feld im Order-Abschnitt (Added on 2010-06-15)

- **Order.DeliveryDate:** Lieferdatum.

Global ID eingeführt (Added on 2010-07-29)

- **<Table>.GlobalID:**Systemübergreifende ID, die in allen PXML-Tabellen.
- **Bar.Ident** ist nun obsolet, und wurde aus dem Standard entfernt.

ProdRot eingeführt (Added on 2010-07-29)

- **Steel.ProdRotX/Y/Z:** Dreh-Empfehlung für Bewehrungsproduktion hinzugefügt.
- **Slab.AngelOfRotation:** ist nun obsolet, und wurde aus dem Standard entfernt.

PTS-Spezifikation (Added on 2010-07-29)

- **Production Test Service:** Rückmeldung vom Prüfsystem.
- **Machine Return:** Rückmeldung von der Maschine.

Klarstellung zum Character-Encoding (Added on 2010-07-29)

- **Character-Encoding:** Es gelten die gängigen XML-Encoding-Regeln.

Erweiterungsvorschläge (Added on 2010-07-29)

- Es wurde ein eigenes Kapitel hinzugefügt, das eventuelle zukünftige Erweiterungen enthält.

Neue MeshType-Werte (Added on 2010-07-29)

- 5 = Deckelmatte 2D; wie Typ '1', wird aber zusammen mit Typ '4' ausgeliefert.
- 6 = Deckelmatte 3D; wie Typ '3', wird aber zusammen mit Typ '4' ausgeliefert.

Neue Felderim Girder-Abschnitt (Added on 2010-08-18)

- **Girder.Width:** Trägerbreite in mm, zwischen den zwei äußersten Punkten.

Neue Felderim Feedback-Abschnitt (Added on 2010-11-25)

- **MaterialWeight:** Gewicht des gemeldeten Materials.

Neue Feld im Slab-Abschnitt (Added on 2011-01-31)

- **ExpositionClass:** Expositionsklasse.

Neue Feld im Bar-Abschnitt (Added on 2011-03-16)

- **ShapeMode:** Darstellungsart der Biegeform.

Mit der Einführung dieses Feldes wurde auch die Thematik der "schematischen" Darstellung überarbeitet. Diese bislang nur unscharf definierte Möglichkeit wurde präzisiert und vereinfacht und hierbei auch an die Realität bestehender Implementierungen angepasst.

Erweiterung der Definition des Steel.MeshType (Added on 2011-04-22)

Neben der eigentlichen Typ-Angabe soll es auch möglich sein, typspezifische Details anzugeben.

PXML-Spezifikation nun auch in Englisch verfügbar (2011-06-03)

Neues Feld im Feedback-abschnitt (Added on 2011-10-04)

- **InfoValue:**Zusätzlicher Informations-Wert.

Neues Feld im DocInfo-Abschnitt (Added on 2012-02-02)

- **ConvertConventions:** Für deklarierte Abweichungen vom PXML-Standard.

Vereinfachung des Multilayer-Konzeptes (2012-02-08)

- **Outline.Layer:**Wurde hinzugefügt.
- **Steel.Layer:**Wurde hinzugefügt.
- **Slab.MasterLevel:**Wurde entfernt.

Parallel- und Reihenschaltung von PTS-Servers (2012-09-14)

Neues Feld im Feedback-Abschnitt (Added on 2012-11-29)

- **Machine:** Maschine oder PTS-Server der die Meldung generiert hat.

Ergänzungen zur Konvertierung von und zu BVBS (2013-06-10)

Erweiterungen für Gitterträger mit variabler Rasterbreite (Added on 2013-11-10)

- **PeriodOffset:** Ist nun nicht mehr auf Werte zwischen 0 und 200 eingeschränkt.
- **Section**: Neue Untertabelle für individuelle Perioden.

Mode-Tabelle hinzugefügt (2013-22-25)

Neues Feld im Product-Abschnitt (Added on 2013-11-27)

- **RotationPosition**.

Version 1.3 (2015-03-01)

In Version 1.3 wurde wichtige Erweiterungen und auch Strukturänderungen eingeführt. Bei Beachtung der Kompatibilitätsempfehlungen ist diese Version aber voll rückwärtskompatibel zu PXML 1.2. Alte Systeme, die für die Version 1.2 gemacht wurden, können daher ohne Änderung auch direkt mit der Version 1.3 arbeiten.

Ersatz der *Legacy Slab Fields* durch entsprechende neue *Product*-Felder

- **Siehe Abschnitt 3.7.7**.

Einführung des *PXML Delegate File*

- **Siehe Abschnitt 1.6**.

Definition des *Referenz-Punktes* eines Elementes

Die Empfehlung keine Cutout-Shapes in Mountparts zu verwenden wurde entfernt

Einführung der 3D-Geometrie für Outlines

- **Neue Shape/SVertex-Felder DX, DY, RefHeight, siehe Abschnitt 3.8.11**.
- **Neue Outline Feld ObjectID, siehe Abschnitt 3.8.10**.

Neues Feld im Order-Abschnitt

- **OrderArea, siehe Abschnitt 3.3.1**Fehler! Verweisquelle konnte nicht gefunden werden..

Neues Tabelle im Product-Abschnitt

- **ElementInfo, siehe Abschnitt 3.6**.

Präzisierung der Regeln für das Zusammensätzen von Doppelwänden und für die Darstellung in Projektkoordinaten.

- **Regeln für das Zusammensätzen von Doppelwänden: siehe Abschnitt 3.5.4.**
- **Darstellung in Projektkoordinaten siehe Abschnitt 3.5.8.**
- **Neues Feld ProductTornMoveX, siehe Abschnitt 3.5.4.**

Version 1.3 – Later Additions

2016-03-24

a) Definition der **Vereinfachten Volumenberechnung**. Siehe Abschnitt 3.8.11.
b) Präzisierung zum *Outline.Name*-Kompatibilität mit UNICAM. Siehe Abschnitt 3.8.3.
c) Korrektur der Include-Merge-Regeln: Unterstrukturen des Delegate-Files dürfen mit Unterstrukturen aus dem Include-File koexistieren. Siehe Abschnitt 1.6.

2016-04-06

a) Tabelle **FbVal** als Untertabelle der *Feedback*-Tabelle eingeführt. Siehe Abschnitt 3.14.4
b) Beispiele für **FbVal** Verwendung hinzugefügt. Siehe Abschnitt 3.14.7
c) Tabelle **ElemInfoVal** als Untertabelle der *ElemInfo*-Tabelle eingeführt. Siehe Abschnitt 3.6.3
d) Neues Feld **F** im *Section*-Eintrag der *Girder*-Tabelle. Siehe Abschnitt 3.11.13.1
e) Neue Gitterträger Typen für **Versaträger** in der *Girder*-Tabelle. Siehe Abschnitt 3.11.5

2017-02-20

a) Neue Mode-Direktiven: **EnableProduction**, **EnableReinforcement**, **EnableProcurement**. Siehe Abschnitt 3.2.5
b) Zusätzliche Felder für die Übergabe der Bauwerksstruktur: **Structure**, **Building**, **SubStorey**. Siehe Abschnitt 3.3.1.
c) Neues Feld **ErpProjectUnit**. Siehe Abschnitt 3.3.1.
d) Neuer *ElementInfo*-Typen **Concrete, ErpProjectUnit, PriceQty, PlanningQty**. Siehe Abschnitt 3.6.2.
e) Einheitenangabe **Unit** im *ElementInfo*. Siehe Abschnitt 3.6.2.
f) Einheitsangabe **U** in *ElemInfoVal*. Siehe Abschnitt 3.6.3.
g) Neue Tabellen OrderInfo und OrderInfoVal. Siehe Abschnitt 3.4
h) Neue Felder in *Girder*-Tabelle: DiagonalWireDiameter (Abschnitt 3.11.4), ArticleNo (Abschnitt 3.11.7)
i) Ergänzung: Empfehlung zur Strukturierung der *StackNo*, siehe Abschnitt 3.5.7.
j) Weitere FbVal-Typen: **Customer**, **Pos, ShiftID, ShiftStart**. Siehe Abschnitt 3.14.4

2017-07-07

a) Neuer FbVal-Wert „**ProdArticle**“ und neues Wr-Feld „**Art**“. Siehe Abschnitt 3.14.4

2018-06-30

a) PTS Kommunikation: TCP/IP-Socket-Kommunikation durch Webservice-Kommunikation ersetzt. Siehe Abschnitt 3.14.8.2
b) Beschreibung des Konzeptes **Abzugskörper** in die Betonmodellierung einzubeziehen. Siehe Abschnitt 3.8.6.
c) Einführung der **Platzierungs-Rotation für Slab**. Siehe Abschnitt 3.7.2.
d) Einführung der **Produktions-Direktiven für Slab**. Siehe Abschnitt 3.7.3
e) Einführung der **Platzierungs-Rotation für Outline**. Siehe Abschnitt 3.8.1.
f) Einführung der vollständigen **Stapel-Geometrie**. Siehe Abschnitt 3.5.7
g) Der Begriff der *Referenzposition* wurde entfernt.
h) *Product.RotationPosition* wurde als obsolet erklärt. Siehe Abschnitt 3.5.6.
i) Komplexe *Include*-Direktiven eingeführt. Siehe Abschnitt 1.6.
j) Einführung neuer *FbVal*-Typen : **CutBegin**, **CutEnd**, **WeldBegin**, **WeldEnd**, **BendBegin**, **BendEnd**, **PlaceBegin**, **PlaceEnd**. Siehe Abschnitt 3.14.4.
k) Einführung neuer *FbVal*-Typen : **Wld**, **Wld_Target, m2**. Siehe Abschnitt 3.14.4.
l) Einführung zusätzlicher Felder der Schweißpunktrückmeldung: **Y**, **WH**. Siehe Abschnitt 3.14.4.2

2018-07-03

a) Einführung der **Darstellung in vereinfachter Geometrie.** Siehe Abschnitt 3.7.8.

2019-02-28

a) Einführung der **D3D-BF2D.** Siehe Abschnitt 3.10.16.16.
b) Einführung von *SVertex*.**Profile**. Siehe Abschnitt 3.8.11.
c) Einführung von *Product*.**StackID**. Siehe Abschnitt 3.5.7.
d) Einführung der **Platzierungs-Rotation für Steel**. Siehe Abschnitt 3.9.1
e) Präzisierung der *Girder.Section* Definition. Siehe Abschnitt 3.11.13.1.

2019-12-19

a) Präzisierungen zur *Girder.Section* Definition. Siehe Abschnitt 3.11.13.
b) Empfehlung zur Generierung von *GlobalIDs* ergänzt. Siehe Abschnitt 3.1.4.
c) Konzept des Virtuellen Elementes eingeführt. Siehe Abschnitt 3.5.2.

2021-01-08

a) Erweiterung der *WeldingPoint*-Definition durch Festlegung von Standard-Werten für den *WeldingPointType* und durch Einführung des neuen Feldes *GroupID*. Siehe Abschnitt 3.10.15.
b) Zusätzliche *FbVal Type*-Definitionen. Siehe Abschnitt 3.14.4.
c) Präzisierung der *SVertex.Profile*-Verwendung für mehrschichtige *Slabs*. Siehe Abschnitt 3.8.11.

2021-05-03

a) Präzisierung der *SVertex.Profile*-Beschreibung Abschnitt 3.8.11.
b) Neues Standard-Tags für *ElementInfo*: **AccAreaAdd, AccAreaExtAdd**, **ArchitecturalPart**, **TransportInfo**. Siehe Abschnitt 3.6.1.
c) Neues Standard-Tags für *ElemInfoVal*: **Name**. Siehe Abschnitt 3.6.3.

2021-06-24

a) Korrektur der *Slab.ProdRotX/Y/Z* Sequenz. Siehe Abschnitt 3.7.3.

2022-11-17

a) Einführung *Slab.ProdX/Y/Z*. Siehe Abschnitt 3.7.3.
b) Neue Standard-Werten für *WeldingPointType*: -23, -29, -30. Siehe Abschnitt 3.10.15.
c) Präzisierungen zur Verwendung von *Steel.Type*, *Steel.MeshType*, *Bar.ReinforcementType*.
d) Neue ProductTypes BM, CL, ST, MD, DT. Siehe Abschnitt 3.5.2

2023-03-23

a) Neue *ProductType*-Wert **InSitu** hinzugefügt. Siehe Abschnitt 3.5.2.
b) Neue *PartType*-Werte **05** hinzugefügt. Siehe Abschnitt 3.7.1.
c) Korrektur eines Dokumentationsfehlers bezüglich der DW-Produktionsdirektiven. Siehe Abschnitt 3.7.4

2024-01-28

a) Neuer *FbVal*-Typ **HltBarTIntvl** hinzugefügt. Siehe Abschnitt 3.14.4.
b) Neuer *WeldingPoint*-Typ **GrippingPoint** hinzugefügt. Siehe Abschnitt 3.10.15.3.
c) Bugfix im Beispiel 3 aus Abschnitt 3.10.16.11.
d) *ObjectID* für Steel eingeführt und Beschreibung zur *ObjectID* verfeinert. Siehe Abschnitte 3.8.10 und 3.9.11

