Publicly Available Specification

ISO/PAS 8329

First edition
2024-08

Extended master connection file (χMCF) — Description of mechanical connections and joints in structural systems

Reference number
ISO/PAS 8329:2024(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 4, *Industrial data*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

This document aims at describing mechanical connections or joints related to mechanical systems or structures. The demand for such a standard has grown from the observation that modern product lifestyle management (PLM) systems, while working well with part information (e.g. geometry, material, weight), are lacking a consistent handling of logical and process related connection information (e.g. parts being connected, orientation of point connections, assembly process parameter).

PLM workstreams need to include connection data to automate development processes and enable seamless data flows between engineering functions. χMCF is intended to be the "language" that is understood and used by the various tools to exchange connection data along the development chain.

The initial motivation to develop this document came from the automotive industry (see Annex C for background and context on this document). However, there is no element in this document that limits it to this industry. It is clearly targeted to support virtual development processes for mechanical systems or structures in any industrial area.

One design goal of χMCF is to support the widest possible range of development and manufacturing processes. This makes it very likely that xMCF and STEP (ISO 10303-242),[1] will be used together. Annex B investigates how this can be done in a way that benefits both standards.

Regardless of the respective industrial domain, complex technical systems (e.g. vehicles, planes, ships) typically consist of thousands of individual parts which are assembled by joints. Depending on the involved materials and the manufacturing processes, a wide range of joining types are used within an individual technical structure or system. Typical connection types are welds, bolt connections, adhesives, rivets, clips, etc. Efficient and reliable data management of such connection data is not only required for the actual design and verification process [computer-aided design (CAD) and computer-aided engineering (CAE)], but also for manufacturing planning and even cost estimation. Various design, material and manufacturing parameters are required to be managed for each connection.

Details for connections or joints grow and mature along the development process. At different development stages (e.g. concept phase, detailed design, verification, manufacturing planning) and engineering functions (e.g. CAD, CAE, manufacturing), data will be added and consumed. Therefore, a database for connection data is required. But also, the software tools adding or extracting data need to understand the data structure and use a common description language. χMCF, defined in this document, serves as this language.

The advantages are evident. Integrating dedicated connection data into the PLM structure and using a common language (χMCF) for data exchange avoids data conversions or re-generations and, therefore, decreases inconsistencies and flaws during system development.

Extended master connection file (χMCF) — Description of mechanical connections and joints in structural systems

1 Scope

This document specifies XML definitions that are used to describe data and information related to connections or joints in mechanical systems or structures.

The following is within the scope of this document:

— description and explanation of XML definitions for logical or process related data or other properties of a connection.

The following aspects are outside the scope of this document:

— geometry of fasteners or other parts,

— handling of χMCF data in

  — product data management (PDM) systems,

  — subscriber data management (SDM) systems, and

  — other data management systems.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

No terms and definitions are listed in this document.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp,

— IEC Electropedia: available at https://www.electropedia.org/.

4 Design principles and basic features of χMCF

4.1 General

The Extended Master Connection File (χMCF) is a container for connection information of complex structures. A complex structure consists of individual parts which are joined together. Connections establish a topology between the parts. Therefore, a database or container designed to gather connection information should be equipped with data structures which reflect this topology between the parts.

χMCF is intended to define an industry standard for the exchange of connection data between different CAx tools (e.g. CAD, CAE, CAM, CAT) along development process steps. Design principles for χMCF are required to keep the standard as lean as possible on one hand, but also enable use case dependent extensions.

Subclause 4.2 explains the design principles and basic features of χMCF.

4.2 Design principles

The design of χMCF is guided by the following principles:

a) χMCF should be able to completely and unambiguously describe all relevant connections/joints that are in use in the automotive or other industries. Amongst others, this includes spot welds, seam welds, rivets, adhesives.

b) χMCF should be able to address all kinds of CAx processes.

c) χMCF contains only information relevant for connections. Hierarchical product structures, assembly sequences, part variants etc. are not the subject of χMCF. Such kind of information needs different methods for propagation. However, χMCF may refer to such "external" information, for example part codes. This principle provides the flexibility to use χMCF in any development process variant established at different companies.

d) χMCF has to be flexible and easy to extend to any future joint types and applications.

e) χMCF is based on the industry standard extensible markup language (XML).[2]

f) Connection data in χMCF must be unique.

g) The content of χMCF data may be incomplete to a certain extent. This addresses the fact that new data is created continuously and needs to be stored throughout the course of CAx processes, without changing its vessel.

h) χMCF follows the max-min principle. It contains information as much as necessary and, at the same time, as little as possible.

i) χMCF shall enable the reconstruction of connections at any certain stage of the involved processes without loss of data or risk of ambiguities.

j) Data in χMCF format shall be kept compact. Elements shall be reused, whenever possible.

k) χMCF offers containers which can be assigned to any certain connector, to a collection of connectors or even to the complete file. This allows incorporation of software or usage specific data before or without standardization.

l) χMCF forms a good candidate for long-term archival of connection data due to its simplicity and extendibility.

XML has been selected as a foundation since it is by itself an industry standard and human readable. XML facilitates efficient data structures which describe the connection topology of such complex structures like automobiles or planes.

4.3 Idealization of joints

Different types of joints have different characteristics. They can differ from each other by their geometrical shapes, mechanical properties like strengths for different loadings, manufacturing processes etc.

To allow for efficient description of joints, some simplifications and idealizations are necessary. The approach chosen by χMCF is to classify joints by their most basic and mandatory attribute, namely its geometrical dimensions. Thus, there are 0-, 1- and 2-dimensional joints in χMCF.

Figure 1 — Seam weld as 1-dimensional joint

A spot weld is treated as a 0-dimensional joint in χMCF. In this way, a (an idealized) spot weld is geometrically described by its coordinate vector *x* and its diameter *d* as an additional attribute. Besides spot welds, there are more joints which can be treated as 0-dimensional.

A seam weld is a typical representative of 1-dimensional joints, see Figure 1 above. It is characterized by a curve describing its spatial course and additional parameters (attributes) determining the sectional shape perpendicular to the curve.

Similarly, adhesive joints can be modelled as 2-dimensional surfaces.

4.4 Reconstruction of joints from χMCF

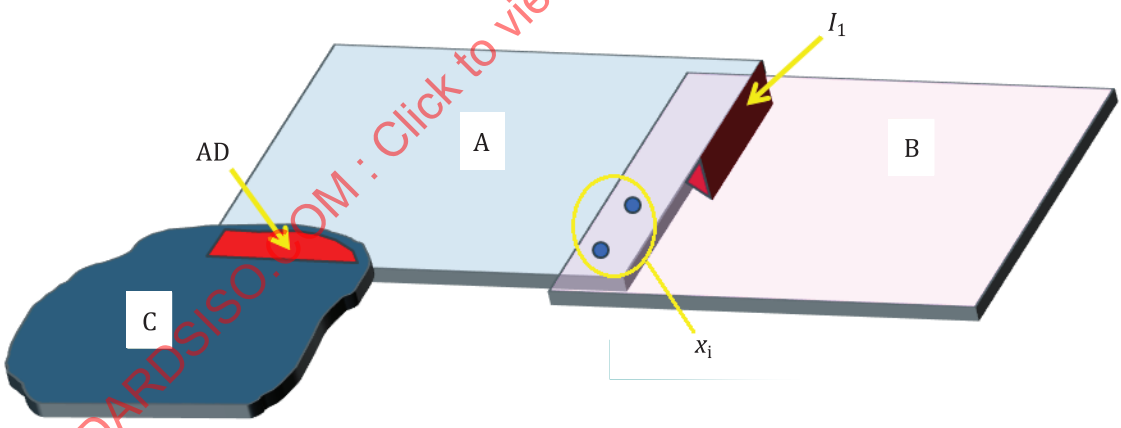
The reconstruction of joints from χMCF is an important use case. It is crucial that it is possible to reconstruct any joint in its idealized form uniquely by means of the introduced parameters and attributes. In case of a spot weld, a unique reconstruction is possible by the coordinate vector *x* and the diameter *d*, plus the sheet thicknesses which by themselves are not a constituent of χMCF (recall χMCF contains only information relevant to joints), but of the corresponding CAD or CAE model.

4.5 Description of topology

As mentioned before, a complex structure arises by connection of parts and sub-structures (assemblies). The connections introduce a topology between the individual components. The following example (see Figure 2) demonstrates the way how χMCF facilitates description of such topology:

— Part (or Assembly) A is joined to Part B by the seam weld 1 along the curve l_1 and the spot welds at positions x_i, and

— Part (or Assembly) A is connected to Part C by the adhesive AD in the area A, etc.



Key

A, B, C	parts
I_1	seam weld 1
x_i	spot welds
AD	adhesive

Figure 2 — Topological relations between parts and assemblies

This kind of topology is represented in χMCF by the element `<connection_group/>`. A `<connection_group/>` comprises all joints which connect the same parts (or assemblies).

Frequently, more than two parts are joined. A spot weld can, for instance, join three sheets, a screw even more. Such situations are covered, too.

According to design principle c), overall product structure cannot be reproduced from χMCF. For example, any of the product structures shown in Figure 3 would equally fit to Figure 2:

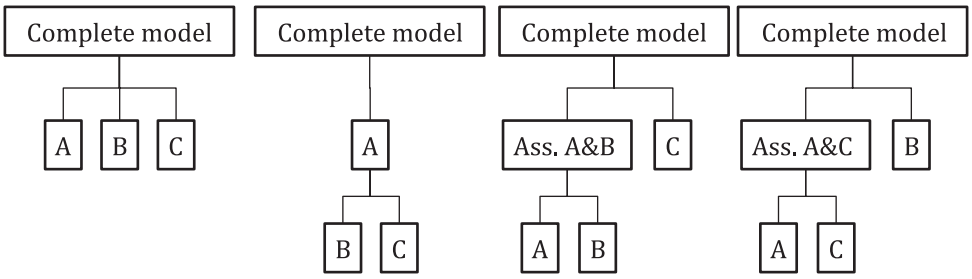


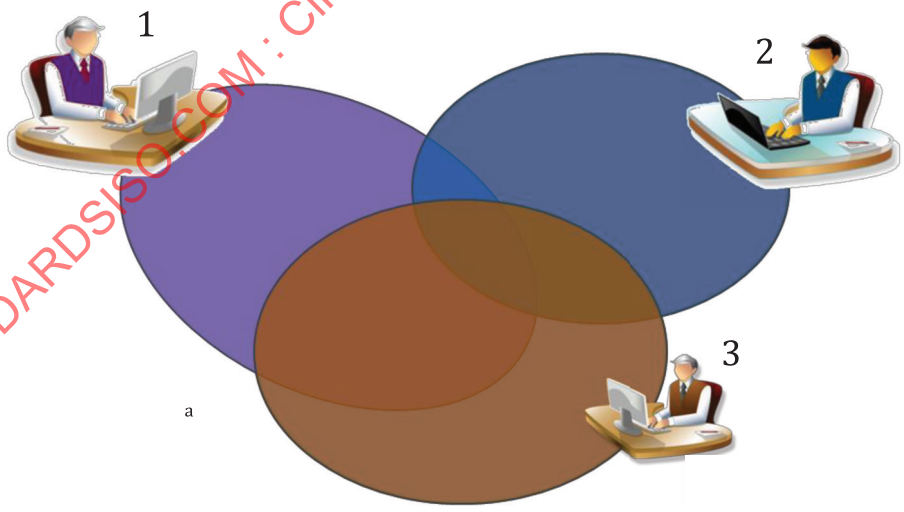
Figure 3 — Product structures fitting to previous figure

NOTE This list of four product structures shown in Figure 3 is not exhaustive.

4.6 χMCF in the development processes

A typical development process is a long chain involving many (maybe overlapping) single steps, e.g. design, construction, prototyping, simulation, testing, production planning (see Figure 4). Depending on the manufacturer considered, information of connections and joints arises at different stages of the process and comes from different parties (see Figure 5). An efficient handling and management of this information can only be guaranteed by a (common) database/container which contains the information uniquely. This shall be guaranteed by using χMCF.

Figure 4 — Development process



Key

1 design, construction
2 engineering
3 production planning
a χMCF.

Figure 5 — χMCF as a platform for connection data in the complete development process

A careful look at Figure 5 provides understanding on how the work with χMCF in a real process can be organized: χMCF is a structured set which can be divided into several overlapping subsets. Each subset contains a part of connection information which is of interest for a certain party, for instance simulation or planning. The intersection of all subsets contains information which is of interest for all the parties involved, e.g. coordinates and flange partners.

As mentioned before, the information contained in χMCF is not necessarily complete, at least not at an early stage of the development process. Rather its content grows while the process is advancing. Defining the individual joint and filling up the container thus builds up a continuous process. As shown in Figure 5, connection information can be created by any of the involved parties (e.g. design, construction, engineering, planning). The common situation is that each party contributes part of the information (e.g. geometrical, technological) defining a specific joint. Merging of the partial information leads to the complete characterization of the joint. Therefore, χMCF is an ideal tool to enable this dynamic process since filling up χMCF means merging information.

Figure 5 also illustrates that connection information (full or partial) is available to all involved parties once it is defined and stored in χMCF. Thus, unnecessary duplication of effort is avoided automatically. Typically, different teams work in different environments using different software tools. Provided all involved systems support χMCF, translation of data from one format to another will not be necessary anymore. This will save development cost and avoid loss of data caused by the translation.

Information contained in χMCF can be used to automate many tasks in a development process and will therefore facilitate efficiency gains:

— Automatic CAE model assembly
  Most FE preprocessors can mesh parts automatically in batch-meshing mode. An automated model assembly can be realized by the connection information contained in χMCF.

— Automatic Programming of Welding Robots
  Based on χMCF, welding robots can be programmed automatically.

An essential feature of χMCF is that it contains only information relevant to the joints. No data are included which are dependent on a specific development process. Therefore, χMCF can be implemented into any development process. Depending on the application, it is possible to use χMCF as a stand-alone database or integrate χMCF into an even more comprehensive database.

5 Keywords of XML specification

5.1 Keywords

As in any XML file, the carrier of information in an χMCF file is an element which can be equipped with some attributes and child elements. Elements and attributes are defined by their names (identifiers) and values (information).

By the XML standard, values assumed by elements can be distinguished by their types, e.g. boolean, float, double, string, date. The same applies to attributes. The user can determine how elements and attributes are used (optional, required or prohibited). If declared necessary, the frequency of occurrence of elements with a given name (number of siblings of identical names) can be restricted. In the XML schema, this is specified by the attributes `minOccurs` and `maxOccurs`.

In accordance with the XML Schema Definition Language version 1.1 Part 2 the following keywords are used in the current document to characterize the elements and attributes:

— Type,

— Value space,

— Default,

— Use,

— Multiplicity (corresponds to the attributes `minOccurs` and `maxOccurs` of the element `<xs:element/>` of the XML schema),

— Restrictions (corresponds to the element `restriction` of XML schema).

NOTE 1 Up to now, only versions 1.0[2] and 1.1[3] of XML exist, where 1.1 is not widely used. Therefore, most systems still create XML 1.0 files (for differences between both versions see http://www.w3.org/TR/xml11/#sec-xml11).

The type of the value of an element or attribute is specified by the key-word `Type`. The numerical ID of a property (attribute `pid`) of a `<part/>` element for instance is an integer, which is a built-in type of XML standard.

Examples for the most common types in XML are:

— `xs:string`,

— `xs:decimal`,

— `xs:integer`,

— `xs:float`,

— `xs:boolean`,

— `xs:date`,

— `xs:time`.

NOTE 2 The maximum number of decimal digits you can specify is 18.

However, only positive integers are usually used in this context. This means that the possible values of the ID (type integer) have to be restricted. To specify the values which are allowed for an element or an attribute, the key-word Value space is used. The Value space can be given as an enumeration (a finite set), or an explicitly defined set. For example, a positive integer is symbolized by > 0 whereas a float between 0,0 and 1,0 is given by [0,0, 1,0], according to mathematical notation.

Some elements and attributes obtain default values if they are not explicitly specified in the χMCF file. The default values to be adopted are defined by the keyword `Default`.

In this document, the special type “alphanumeric” is frequently used for labels of parts and assemblies, which deserves a careful discussion. In the CAD world, a label is synonymous with the name of a part, a geometric object etc. Not only letters “[A-Za-z]”, but also numbers “[0-9]” and other special characters such as “[-.$#±]” and more are used for labels. Sometimes, the first character is restricted to “[A-Za-z]”. Thus, it is difficult to give an exact definition for the type “alphanumeric” which would fit to the individual need. Fortunately, when using XML’s “encoding” attribute, even non-ASCII characters can be handled easily, e.g. Arabic, Chinese, Cyrillic, Greek, Hebrew. Nevertheless, labels should not start or end with white space.

The key-word `Use` specifies, whether an element or an attribute is optional, required or prohibited. The frequency of the occurrence of an element or attribute is defined by `Multiplicity`, that is in the form: `minOccurs` ≤ `Multiplicity` ≤ `maxOccurs`. By convention, when `Use` is optional, `minOccurs` is 0. Any additional restrictions imposed on an element, or an attribute are specified by the key-word `Restrictions`.

As explained above, the individual use of some elements or attributes may be optional. But some of them must be coherent (thus redundant in certain sense). For instance, the `label`, numerical ID of a property (`PID`), and alphanumerical name of a property (`pname`) of a part or an assembly represent the same part (except for e.g. tailored blanks) and one can use one or the other or both to identify a part.

6 Parts, properties and assemblies

6.1 General

χMCF describes how parts, properties and assemblies are connected by joints in a pre-defined way. Therefore, a clear understanding is needed about what a part, property or assembly is, in our context.

6.2 Parts

6.2.1 General

Parts are logical groupings of 3D objects. Their objective is to provide a general nomenclature of the pieces which form a certain product. This nomenclature allows communications between all stakeholders of all involved processes.

Typically, it is assumed that parts do not disintegrate into several physical compounds.

Parts can be instantiated at different locations of a product, e.g. wheels in a car.

Parts can be mirrored on a symmetry plane of the model, e.g. front doors of a car.

Parts can contain other parts (sub-parts): A car, for example, is made of body in white, power train, doors etc. A door is made of an outer sheet, an inner sheet, a window with its mechanics, some crash reinforcements etc. The mechanics of a window are made of some guiding rails, an electric motor and so on.

Therefore, in the sense of graph theory, parts form a tree (if their instances are considered) or a directed, cycle free graph. Parts without sub-parts are called the "leaves" of this tree or graph.

If a part is mentioned in a list, not only its own content (e.g. finite elements) is addressed, but also all contents of its sub-parts and their children, down to the lowest level (leaves) of the part graph.

6.2.2 Part labels

A part is uniquely identified by its label, up to ditto-parts. Connectors within a connection group that refer to ditto parts shall be able to "detect" the "correct" part instance according to their respective geometrical location.

We assume that mirror parts have other part labels than their "base" parts.

NOTE In most CAx processes, parts have two string attributes: One label describing the name and usage of a part in a human readable form, and another one used for indexing this item in the OEM's "part store". The latter one typically consists of only a few characters (e.g. some 8 to 12), resembles more to a number than to a name, and therefore, is not human readable. In our context, the term "part label" refers to the latter one.

6.2.3 Part instances

Instances of parts, also known as ditto-parts, typically have the same label as their "base" parts. Stating their instance makes such parts uniquely distinguishable, without resort to their geometrical location. Stating an instance without a part is meaningless, however.

6.3 Properties

In CAE, properties are a concept for assigning physical behaviour to several finite elements. Therefore, any finite element can have at most one property. However, frequently there are elements without such properties (e.g. RBEs, masses). In most solvers, properties are uniquely identified by positive integers, so-called property IDs or short: `PID`s. Some other solvers identify properties by an alphanumerical name, short `pname`.

Even if finite elements of different parts have the same physical behaviour (e.g. left and right wing of a car), they usually have assigned different properties. This can be seen as a reminiscence to ancient times when parts had not been invented and properties were also used for administrative purposes.

However, for χMCF, properties are just alternative, non-recursive means for addressing collections of elements.

One specific part often consists of one specific property, only. However, there are important exceptions:

— A tailored blank is a metal sheet which consists of several pieces of simple sheets joined together. The thicknesses and the materials of the individual sheets may both differ. Nevertheless, a tailored blank is one single part from the χMCF point of view. Since one property would not provide an identifier for the complete part, the part label must be used, or else an assembly of several properties.

— Sometimes, a cast part can be represented with finite element (FE) shell element formulation in its thin areas, whereas solid elements (with different properties) are used in other areas.

— Due to stamping processes for example, physical behaviour and thickness can vary even within one originally homogeneous sheet metal, requiring several properties for a correct simulation.

— Occasionally, CAD parts containing several subparts with their properties are aggregated to one single CAE part, consequently still containing several properties.

6.4 Assemblies

In many CAx systems, parts containing sub-parts are called assemblies. The notion distinguishes them from leaves of the part tree or graph.

However, in χMCF, an assembly is considered a set of parts, denoted by their part labels or properties. They do not need to have any special relation respective to the part graph. The opposite is true: χMCF-assemblies address situations where specifying a single property would not address enough, a high-level part would address way too many elements and medium-size parts would not make the job.

On the other hand, this does not happen too often. For example, if a seam weld crosses property boundaries, these properties usually belong to the same tailored blank, therefore, the same part. If there would be a physical gap between the properties, welding would be applied to a single sheet across this gap, which causes new questions to the welding process, as depicted in Figure 6:

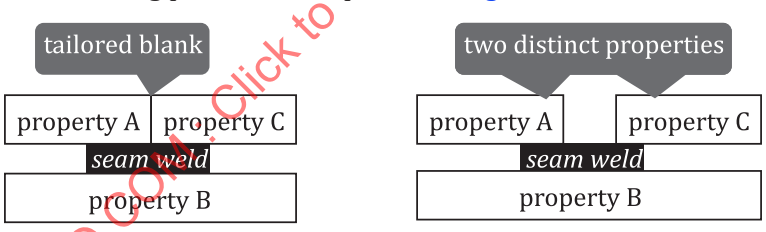


Figure 6 — Seam weld crossing tailored blank versus seam weld crossing physical gap

Even if there is a gap, due to geometrical proximity and usual assembly processes, it is very likely that properties A and C belong to the same part, just one level above, even if there is a gap in part of the graph.

7 File structure of χMCF

7.1 General

As mentioned, χMCF is built upon XML. This eases χMCF to offer a clear logical structure.

The root/document element of χMCF must be named `<xmcf/>`. The root element may contain the following types of child elements:

a) Comments following the usual XML standard; therefore, not further discussed here,

b) Elements containing general information,

c) Groups of connection specific elements <connection_group/> of arbitrary number,

d) Element <appdata/> containing specific data for individual applications,

e) Element <femdata/> containing finite element specific data.

7.2 Elements containing general information

7.2.1 General

χMCF has the following elements for general information:

— <date/> optional,

— <version/> mandatory,

— <units/> optional.

The root element <xmcf/> contains the following nested elemenths shown in Table 1.

Table 1 — Nested elements of element <xmcf/>

Nested elements	Multiplicity	Use	Constraint / Remarks
date	1	Optional	-
version	1	Required	-
units	1	Optional	-
appdata	1-*	Optional	See 7.3.2
femdata	1-*	Optional	See 7.3.3
connection_group	1-*	Optional	See 7.4

7.2.2 Date

The element <date/> of the format “yyyy-mm-dd” specifies the date on which the file was created. It follows the ISO 8601 series.[4]

EXAMPLE

```
<?xml version="1.0" encoding="UTF-8" ?>
<xmcf xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="xmcf_3_1_1.xsd">
    <date> 2023-04-13 </date>
    <version> 3.1.1 </version>
    <units length="mm" angle="rad" mass="kg" force="N" time="s"/>
    ...
</xmcf>
```

7.2.3 Time

The element <time/> of the format “hh:mm:ss±hh:mm” specifies the time on which the file was created. It follows the ISO 8601 series [4].

Time element may exist only if date element exists.

EXAMPLE

```
<?xml version="1.0" encoding="UTF-8" ?>
<xmcf xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="xmcf_3_1_1.xsd">
    <date> 2023-04-13 </date>
    <time> 15:34:05-01:00 </time>
    <version> 3.1.1 </version>
```

```
    <units length="mm" angle="rad" mass="kg" force="N" time="s"/>
    ...
</xmcf>
```

7.2.4 Version

The code of the χMCF standard version on which the current file is based is specified by the element `<version/>`.

The version code of χMCF files following this document is version 3.1.1.

EXAMPLE

```
<?xml version="1.0" encoding="UTF-8" ?>
<xmcf xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="xmcf_3_1_1.xsd">
    <date> 2023-04-13 </date>
    <time> 15:34:05-01:00 </time>
    <version> 3.1.1 </version>
    <units length="mm" angle="rad" mass="kg" force="N" time="s"/>
    ...
</xmcf>
```

7.2.5 Unit system

The unit system used by χMCF is based on the International System of Units (SI)[5] and is represented by the element `<units/>`. Both the base and the derived units are supported, including decimal prefixes.

Following non-SI units are allowed (in addition): Length [in] and [ft]; Mass [lb], see Reference [6].

No units need to be specified for dimensionless physical quantities, e.g. friction coefficients.

The XML-specification of `<units/>` is shown in Table 2:

Table 2 — XML-specification of `<units/>`

Attribute	Use	Value space	Default
length	Optional	“mm”, “m”, “in”, “ft"	“mm”
angle	Optional	“deg”, “rad”	"deg”
mass	Optional	“g”, “kg”, “t”, “lb”	“kg”
force	Optional	“kN”, “N”	“N”
time	Optional	“s”, “min”, “h”	“s”
torque	Optional	“Nm”	“Nm”
angular_speed	Optional	“rad/s”, “Hz”, “kHz”, “rpm”	“Hz”

EXAMPLE

```
<?xml version="1.0" encoding="UTF-8" ?>
<xmcf xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="xmcf_3_1_1.xsd">
    <date> 2023-04-13 </date>
    <version> 3.1.1 </version>
    <units length="mm" angle="rad" mass="kg" force="N" time="s"/>
    ...
</xmcf>
```

7.3 Application, user and process specific data

7.3.1 General

The user/application software can store additional information into a χMCF file. In this way, flexibility is created that allows easy integration of χMCF into an existing development process.

The current χMCF definition allows two such data elements:

— `<appdata/>`
Contents must be documented by the corresponding application or user. It is not an official part of the χMCF standard.

— `<femdata/>`
Contents must be documented in FATXML[7] and therefore does not need to be described here.

7.3.2 User specific data `<appdata/>`

`<appdata/>` is suitable for any user/application specific information and can be placed on root level (directly within `<xmcf/>` element) and/or within any single connector (elements `<connection_0d/>`, `<connection_1d/>`, and `<connection_2d/>`). Additionally, it can be placed directly under the `<connection_group/>` element.

`<appdata/>` shall contain at least one nested element named after the application or user that is intended to interpret the data. In examples A and B, the associated application is MEDINA, so the nested element is `<MEDINA/>`.

Content of `<appdata/>` is regarded to be “private property” of the corresponding application. However, in terms of best practices, the following is recommended, but not required:

— to place application specific elements into a separate namespace;

— to provide an XML schema for its content.

The user shall be aware that different systems are likely to introduce the same physical parameter at the same time (inducted e.g. by a certain new emerging connecting method) but describe it in their own XML schemas with different element/attribute names.

In this situation, a preprocessor does not have any chance to detect these equivalent parameters. Therefore, it cannot prevent contradictions between different `<appdata/>` blocks of the same χMCF file.

EXAMPLE 1 `<appdata/>` for MEDINA at root level

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<xmcf xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xmlns:MEDINA="http://servicenet.t-systems.com/medina/xMCF"
xsi:schemaLocation="http://servicenet.t-systems.com/medina/xMCF   mcf_MEDINA.xsd"
xsi:noNamespaceSchemaLocation="xmcf_3_1_1.xsd">
    <date> 2014-08-07 </date>
    <version> 3.1.1 </version>
    <units length="mm" angle="rad" mass="kg" force="N" time="s"/>
    <appdata>
        <MEDINA xmlns="http://servicenet.t-systems.com/medina/xMCF">
            <data_at_root>
                <version MEDINA="MEDINA 8.4.2 Maintenance Release (64 Bit)"/>
                ...
            </data_at_root>
        </MEDINA>
    </appdata>
    ...
</xmcf>
```

EXAMPLE 2 `<appdata/>` for MEDINA at connection level

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<xmcf xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xmlns:MEDINA="http://servicenet.t-systems.com/medina/xMCF"
xsi:schemaLocation="http://servicenet.t-systems.com/medina/xMCF mcf_MEDINA.xsd"
xsi:noNamespaceSchemaLocation="xmcf_3_1_1.xsd">
    <date> 2014-08-07 </date>
    <version> 3.1.1 </version>
    <units length="mm" angle="rad" mass="kg" force="N" time="s"/>
    ...
    <connection_group id="1">
```

```
        <connected_to>
            ...
        </connected_to>
        <connection_list>
            <connection_1d>
                <loc_list>
                    ...
                </loc_list>
                <seamweld>
                    ...
                </seamweld>
                <appdata>
                    <MEDINA xmlns="http://servicenet.t-systems.com/medina/xMCF">
                        <data_at_connector>
                            ....
                        </data_at_connector>
                    </MEDINA>
                </appdata>
            </connection_1d>
        </connection_list>
    </connection_group>
</xmcf>
```

7.3.3 Finite element specific data <femdata/>

For numerical simulation by the finite element method, a joint can be discretized (realized) in different ways depending on the focus of the simulation (e.g. crash, fatigue). It is therefore often necessary to switch from one realization to another one. For this purpose, details of a specific realization can be of interest.

The optional <femdata/> information can be placed within any single connector (relevant elements are <connection_0d/>, <connection_1d/> and <connection_2d/>).

χMCF versions 3.1 or later allow to contain <femdata/> at root level, but this is not allowed in V 3.0 and below. <femdata/> is not allowed on <connection_group/> level in any case.

<femdata/> refers to FE entities that are related to the connector in which <femdata/> is placed. Its content and the referenced elements are specific to a particular solver.

Usually, this kind of referencing is done by solver specific entity IDs, which have no meaning outside the context of a specific finite element model. If, for example, element IDs in this model get renumbered, a χMCF file referencing such element IDs becomes detached and needs to be re-created.

In conclusion, a χMCF file containing <femdata/> always refers to one specific solver deck.

Solver names should be taken from the current FATXML version. Examples are the following:

— PAM-CRASH,

— LS-DYNA,

— RADIOSS,

— OPTISTRUCT,

— NASTRAN,

— PERMAS,

— ABAQUS.

Only <entity/> (see Table 3) is allowed as a nested element of the child element of <femdata/>. Its definition and documentation follow <ENTITY/>, the corresponding element in FATXML.[7]

Table 3 — Nested elements of the child element of `<femdata/>`

Nested elements	Multiplicity	Use	Constraint / Remarks
entity	1-*	Required	Corresponds to element **<ENTITY/>**, defined in Reference [7].

For further definition of `ENTITY` see the document source website for FATXML.[7]

EXAMPLE <femdata/> within a <connection_0d/> element

```
<connection_0d>
    ...
    <femdata>
       <NASTRAN>
            <entity>
                <TYPE>
                    CQUAD
                </TYPE>
                <ID>
                    12345-12356
                </ID>
            </entity>
       </NASTRAN>
    </femdata>
    ...
</connection_0d>
```

Like FATXML, χMCF data can be embedded into solver decks by this means: Any receiving system can easily detect and remove discretization objects, created by a sending system, in order to substitute them by its own new discretization objects.

The `<femdata/>` element can be used versatile for different use cases – even for yet unknown ones. This makes it difficult to define exact semantics.

Specific agreements, for example between preprocessor and solver/postprocessor, may be made to support specific use cases.

7.4 Connection data `<connection_group/>`

7.4.1 General

`<connection_group/>` contains the topological information about the parts or assemblies involved (Clause 6), respectively. As explained in 4.5, joints are grouped together by the parts or assemblies which they commonly connect.

The topological relation (relation of neighbours) is defined by the child element `<connected_to/>` whereas all involved joints are listed in the child element `<connection_list/>` according to their types (see 4.3).

Each `<connection_group/>` is uniquely identified by a numeric identifier (`id`).

NOTE Therefore, χMCF files cannot be simply “pasted together” by use of a standard text editor.

XML-specification of `<connection_group/>` is shown in Table 4 and Table 5.

Table 4 — Attributes of element `<connection_group/>`

Attributes	Type	Use	Constraint / Remark
id	Integer	Required	unique within a χMCF file

Table 5 — Nested elements of element `<connection_group/>`

Nested elements	Multiplicity	Use	Constraint
connected_to	1	Required	-
connection_list	1	Required	-
contact_list	1	Optional	-

An empty or missing `<connected_to/>` element means a connection according to geometric neighbourhood, alone. However, if `<connected_to/>` is present, it must be complete, i.e. no additional connection partners shall be searched. Searching for a geometric neighbourhood can yield different results, depending on the algorithm employed. To avoid ambiguities, no connections with missing `<connected_to/>` should reach the solver. Therefore, `<connected_to/>` should be filled by the pre-processor.

In addition to parts and properties, no other means (such as sets) for grouping objects are allowed.

7.4.2 Connected objects

7.4.2.1 General

The basic objects which can be jointed are parts and assemblies (see Clause 6) which appear as nested elements `<part/>` and `<assy/>` of `<connected_to/>`. The XML-specification of `<connected_to/>` is shown in Table 6.

Table 6 — Nested elements of `<connected_to/>`

Nested elements	Multiplicity	Use	Constraint
part	1 - *	Optional	-
assy	1 - *	Optional	-

7.4.2.2 Element `<part/>`

In χMCF, a part may refer to one CAx part or one CAE property.

A part is described by the element `<part/>` and a numeric `index`, a `label` (part code), a `pid` (property id) or `pname` (property name), all provided as attributes. However, if both attributes “label” and “pid” or “label” and “pname” are present, the label governs.

Although most solvers use numbers as identifiers, ABAQUS uses names as identifiers. To identify a property, only one of `pid` or `pname` is sufficient. If both identifiers are present, they must be equivalent in the sense that they both address the same collection of elements. Rationale for allowing presence of both identifiers is the case that the same mesh, and therefore, same properties are used in both, NASTRAN and ABAQUS. In this situation, it is recommended to have a χMCF file which contains both, PIDs and property names. On the solver side, this would cause no confusion since NASTRAN would ignore the property name and ABAQUS the PID. The responsibility to keep both primary keys unique and equivalent resides on the pre-processor side. Upon import of χMCF to a pre-processor, inconsistent property keys shall cause an error.

The `index` needs to be unique only within the parent element `<connected_to/>`. For specific connections, it is used as the matching index for the base sheet.

The attribute `index` of `<part/>` element is required only if the `part` element is used as a nested element under the `<connected_to/>` element. If the `<part/>` element is used within the element `<assy/>`, then `index` is not allowed as an attribute of the `<part/>` element. The XML-specification of `<part/>` is shown in Table 7.

Table 7 — Attributes of element `<part/>`

Attributes	Type	Value space	Use	Constraint
index	Integer	> 0	Required	Unique and required only within the parent element `<connected_to/>`
label	Alphanumeric	Alphanumeric	Optional	At least `label`, `pid`, or `pname` must exist.
pid	Integer	> 0	Optional	
pname	Alphanumeric	Alphanumeric	Optional	
instance	Alphanumeric	non-empty	Optional	`label` must exist if `instance` is used.

EXAMPLE 1 <part/> with required attributes only (pid or pname may be used alternatively to label)

```
<connected_to>
    <part index="1" label="PART_7000400"/>
</connected_to>
```

EXAMPLE 2 <part/> with optional use of label and pid

```
<connected_to>
    <part index="1" label="PART_7000400" pid="3202132"/>
</connected_to>
```

EXAMPLE 3 <part/> with pname to identify a part or property

```
<connected_to>
    <part index="1" pname="P3202132 Thin Shell Property"/>
</connected_to>
```

EXAMPLE 4 <part/> using a label and an instance to identify a part

```
<connected_to>
    <part index="1" label="PART_WHEEL_900" instance="4"/>
</connected_to>
```

7.4.2.3 Element `<assy/>`

An assembly represents a sub-structure consisting of at least two `<part/>` elements. It is described by the element `<assy/>` with only the mandatory attribute `index`. The XML specification of element `<assy/>` is shown in Table 8.

Table 8 — Attributes of element `<assy/>`

Attributes	Type	Use	Constraint
index	Integer	Required	Unique within the parent element

In the following examples, any `pid` attribute can be supplemented or replaced by a `pname` attribute.

EXAMPLE 1 Full definition of `<assy/>` element within `<connected_to/>`

```
<connected_to>
    <assy index="42">
        <part label="PART_7000400" pid="110013"/>
        <part label="PART_7000800" pid="110099"/>
    </assy>
</connected_to>
```

EXAMPLE 2 Full definition of the combined use of `<part/>` and `<assy/>` elements within `<connected_to/>`

```
<connected_to>
    <part index="1" label="PART_9004400" pid="3202132"/>
    <assy index="42">
        <part label="PART_7000400" pid="110013"/>
        <part label="PART_7000800" pid="110099"/>
```

```
        </assy>
    </connected_to>
```

EXAMPLE 3 Usage of the attribute instance for a <part/> within an <assy/> element

```
    <connected_to>
        <part index="1" label="PART_9004400" pid="3202132"/>
        <assy index="42">
            <part label="PART_7000400" instance="2" pid="110013"/>
            <part label="PART_7000800" pid="110099"/>
        </assy>
    </connected_to>
```

EXAMPLE 4 Minimum definition for the combined use of <part/> and <assy/> elements within <connected_to/>

```
    <connected_to>
        <part index="1" label="PART_9004400"/>
        <assy index="42">
            <part label="PART_7000400"/>
            <part label="PART_7000800"/>
        </assy>
    </connected_to>
```

or

```
    <connected_to>
        <part index="1" pid="3202132"/>
        <assy index="42">
            <part pid="110013"/>
            <part pid="110099"/>
        </assy>
    </connected_to>
```

The body of an <assy/> element equals that of a <connected_to/> element. But the meaning is different: All parts within one <assy/> element are meant to constitute the same side/layer/partner of a flange, whereas all members of a <connected_to/> element are different sides/layers/partners of a flange.

Recursion, such as an <assy/> element nested within another <assy/> element, is not allowed.

7.4.2.4 Special topological situations

7.4.2.4.1 Stacking

The aim of the <connection_group/> element is to group up all the joints that connect the same parts.

Therefore, <connected_to/> contains each part connected in the joint only once. However, in some situations it can be important to explicitly define in which order some parts of the group are connected.

This includes the following scenarios:

— the stacking order of the connected parts can be important,

— some parts may be involved more than once in the same joint (self-connected joint), where either:

  — each part involved in a self-connected joint more than once is known individually, or

  — just the number of parts involved in a self-connected joint is known, or

  — a combination of the two sub-scenarios above.

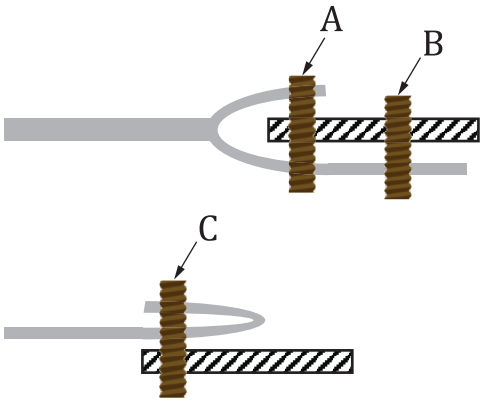


Figure 7 — Special stacking topologies

In Figure 7, all joints, A, B, C, exist within the same `<connection_group/>`, but each joint is connected in a different way.

```
<connection_group>
    <connected_to>
        <part index="1" label="PART_7000800"/>   <!-- grey part in figure -->
        <part index="2" label="PART_7000400"/>   <!-- hatched part in figure -->
    </connected_to>
</connection_group>
```

For joints A and C, the number of flanges connected is more than the number of parts in `<connected_to/>`. Between joints A and C, the flanges feature the same parts, but in a different order.

To store this information for each case, the `<stacking/>` element is used.

7.4.2.4.2 Element <stacking/>

`<stacking/>` may dictate the list of flanges/sheets involved in a joint, as well as their order. Alternatively, `<stacking/>` may indicate the number of flanges/sheets of a joint, without defining which parts are connected more than once. Table 9 shows the nesting element of `<stacking/>` and Table 10 its attribute.

Table 9 — Nested elements of `<stacking/>`

Nested Elements	Multiplicity	Use	Constraint
level	1 - *	Optional	-

Table 10 — Attributes of `<stacking/>`

Attributes	Type	Value space	Use	Constraints / Remarks
nr_levels	Integer	> 0	Optional	if `nr_levels` exists, no `<level/>` elements are allowed in `<stacking/>`. `nr_levels` has to be greater than the number of nested elements of `<connected_to/>`

The attribute `nr_levels` dictates the number of flanges/sheets connected by the joint.

The element `<level/>` within `<stacking/>` is specified as shown in Table 11.

Table 11 — Attributes of `<level/>`

Attributes	Type	Value space	Use	Constraint
order	Integer	> 0	Required	Unique only within the parent element `<stacking/>`
part_index	Integer		Required	

The attribute meanings are:

— `part_index`: The flange partner with this index (see 7.4.2.2). The part of the flange is referenced by the attribute `index` inside the element `<part/>` or `<assy/>` of the `<connected_to/>` element,

— `order`: indicates the position of a flange relative to other flanges.

The order of the levels in the stacking list is identified by the numerical value of their attribute `order`, in ascending order. Therefore, indices must be unique within one stacking list. The restriction that `nr_levels` must be greater than the number of nested elements of `<connected_to/>` implies that `nr_levels` attribute can only be used for self-connected joints.

EXAMPLE 1 The situations in Figure 7 can be described by using `<level/>` elements in order to explicitly define the stacking of the part flanges involved.

```
<connection_group>

    <connected_to>
        <part index="1" label="PART_7000800"/>          <!-- grey part in figure -->
        <part index="2" label="PART_7000400"/>          <!-- hatched part in figure  -->
    </connected_to>

   <connection_list>
        <connection_0d label="A">
            <stacking>
                <level order="1" part_index="1"/>   <!-- grey part in figure -->
                <level order="2" part_index="2"/>   <!-- hatched part in figure  -->
                <level order="3" part_index="1"/>   <!-- grey part in figure -->
            </stacking>
            ...
        </connection_0d>

        <connection_0d label="B">
            <stacking>
                <level order="1" part_index="2"/>   <!-- hatched part in figure  -->
                <level order="2" part_index="1"/>   <!-- grey part in figure -->
            </stacking>
            ...
        </connection_0d>

        <connection_0d label="C">
            <stacking>
                <level order="1" part_index="1"/>   <!-- grey part in figure -->
                <level order="2" part_index="1"/>   <!-- grey part in figure -->
                <level order="3" part_index="2"/>   <!-- hatched part in figure  -->
            </stacking>
            ...
        </connection_0d>
   </connection_list >

</connection_group>
```

EXAMPLE 2 The same situations can also be expressed using the nr_levels attribute, which simply states how many flanges of the <connected_to> parts are involved in each joint.

```
<connection_group>

    <connected_to>
        <part index="1" label="PART_7000800"/>   <!-- grey part in figure -->
        <part index="2" label="PART_7000400"/>   <!-- hatched part in figure  -->
    </connected_to>
```

```
    <connection_list>
        <connection_0d label="A">
            <stacking nr_levels="3"/>   <!-- "hatched", "grey" and one of "hatched"/"grey"
-->
            ...
        </connection_0d>

        <connection_0d label="B">
            ...                              <!-- "hatched", "grey" in any order -->
        </connection_0d>

        <connection_0d label="C">
            <stacking nr_levels="3"/>   <!-- "hatched", "grey" and one of "hatched"/"grey"
-->
            ...
        </connection_0d>
    </connection_list >

  </connection_group>
```

7.4.3 Contacts and friction

7.4.3.1 General

For many joint types, e.g. bolts and screws, friction between the jointed partners plays an important role for the manufacturing and the mechanical behaviour of the joints in service.

In general, friction is a property of pairs of materials in contact. Normally it can be assumed that the friction property, here simply characterized by the static and kinetic friction coefficients, is homogenous. Nevertheless, friction properties shall allow for local modification of an individual connection to enhance the service behaviour.

In χMCF, friction coefficients for any combination of joint partners defined in `<connected_to/>` can be specified by the element `<contact/>` which is nested in the element `<contact_list/>`. Each part in contact is given by the element `<partner/>`. The static and kinetic friction coefficients are defined by the element `<coefficients/>`.

The friction property between the head of a bolt to jointed parts is specified, where the joint is defined.

7.4.3.2 Element `<contact_list/>`

Relevant contacts, which are possible between the flange partners of a `<connection_group/>`, are collected in a `<contact_list/>`. The XML specification of `<contact_list/>` element is shown in Table 12.

Table 12 — Nested elements of element `<contact_list/>`

Nested elements	Multiplicity	Use	Constraint
contact	1 - *	Required	Any set (= non-ordered pair) of physical contact partners is not allowed to appear more than once within a `<contact_list/>`.

The element `<contact_list/>` does not allow for any attributes.

7.4.3.3 Element `<contact/>`

The features or coefficients of a physical contact between flange partners are described by an element `<contact/>`. The XML specification of the element `<contact/>` is shown in Table 13.

Table 13 — Nested elements of element `<contact/>`

Nested elements	Multiplicity	Use	Constraint
partner	2	Required	-
coefficients	1	Required	-

Ordering of `<contact/>` elements within a `<contact_list/>` is irrelevant, since it is assumed that features of a physical contact are invariant under permutation of the two involved materials.

The element `<contact/>` does not allow for any attributes.

7.4.3.4 Element `<partner/>`

Each joint partner involved in a contact is specified by the element `<partner/>`. Only the first level of parts/assemblies which are listed in `<connected_to/>`, is allowed. The XML specification of the `<partner/>` element is shown in Table 14.

Table 14 — Attributes of element `<partner/>`

Attributes	Type	Value space	Use	Constraints / Remarks
part_index	Integer		Required	

The attribute has following meaning:

— `part_index`: The flange partner with this index (see 7.4.2.2). The part of the flange is referenced by the attribute `index` inside the element `<part/>` or `<assy/>` of the `<connected_to/>` element.

The element `<partner/>` does not allow for any nested elements.

7.4.3.5 Element `<coefficients/>`

Static and kinetic friction coefficients are defined by the attributes `static_friction` and `kinetic_friction` of an element `<coefficients/>`, respectively.

Example

```
<connected_to>
    <part index="1" label="PART_9004400" pid="3202132"/>
    <assy index="42">
        <part label="PART_7000400" pid="110013"/>
        <part label="PART_7000800" pid="110099"/>
    </assy>
</connected_to>
<contact_list>
    <contact>
       <partner part_index="1"/>
       <partner part_index="42"/>
       <coefficients static_friction="0.3" kinetic_friction=".25"/>
    </contact>
</contact_list>
```

The element `<coefficients/>` does not allow for any nested elements.

7.4.3.6 Local contact properties

If necessary, local contact properties can be given within any element `<connection_0d/>` or `<connection_1d/>`, respectively (see 7.4.4). In case of conflict, a local `<contact_list/>` overrules the global one.

NOTE `<connection_2d/>` is not relevant for the currently known use cases and was therefore intentionally not included in the list.

XML-specification of `<coefficients/>` are shown in Table 15.

Table 15 — Attributes of element `<coefficients/>`

Attributes	Type	Value space	Use	Constraints / Remarks
static_friction	Floating point	[0, ∞[	Optional	-
kinetic_friction	Floating point	[0, ∞[	Optional	-

A `<connection_list/>` is not allowed to be empty, i.e. at least one connection must be defined.

7.4.4 Joints

All the joints which connect the same set of objects (order does not matter) described in the element `<connected_to/>` are listed in the element `<connection_list/>`. There should be only one connection group for any distinct set of objects in a χMCF file.

As discussed in 4.3, χMCF differs between 0-, 1- and 2-dimensional joints which will be specified in detail in the following clauses. Thus, an element `<connection_list/>` can comprise child elements `<connection_0d>`, `<connection_1d>` and `<connection_2d>` of arbitrary repetitions.

The XML-specification of `<connection_list/>` is given in Table 16:

Table 16 — Nested elements of element `<connection_list/>`

Nested elements	Multiplicity	Use	Constraint
connection_0d	*	optional	-
connection_1d	*	optional	-
connection_2d	*	optional	-

A `<connection_list/>` is not allowed to be empty, i.e. at least one connection must be defined.

7.5 Minimalistic example of a χMCF file

The following example shows how a χMCF file should look.

EXAMPLE

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<xmcf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:MEDINA="http://servicenet.t-systems.com/medina/xMCF"
xsi:schemaLocation="http://servicenet.t-systems.com/medina/xMCF mcf_MEDINA.xsd"
xsi:noNamespaceSchemaLocation="xmcf_3_1_1.xsd">
    <!-- some comments -->
    <date> 2016-01-11 </date>
    <version> 3.1.1 </version>
    <units length="mm" angle="rad" mass="kg" force="N" time="s"/>
    <appdata> <!--appdata at root level -->
        <MEDINA xmlns="http://servicenet.t-systems.com/medina/xMCF">
            <data_at_root>
                <version MEDINA="MEDINA 8.4.2 Maintenance Release (64 Bit)"/>
                ...
            </data_at_root>
        </MEDINA>
    </appdata>
        ...
    <connection_group ...>
        <connected_to>
            <part index="1" label="PART_8000880" pid="20123213"/>
            <part index="2" label="PART_8100340" pid="90123213"/>
        </connected_to>
        <appdata> <!--appdata at connection_group level -->
            <MEDINA xmlns="http://servicenet.t-systems.com/medina/xMCF">
                <data_at_connection_group>
                    ...
                </data_at_connection_group>
            </MEDINA>
```

```
        </appdata>
        <connection_list>
            <connection_0d>
                <femdata>
                    <NASTRAN>
                        ...
                    </NASTRAN>
                </femdata>
               ...
            </connection_0d>
            <connection_1d>
                <loc_list>
                    ...
                </loc_list>
                <seamweld>
                    ...
                </seamweld>
                <appdata>
                    <MEDINA xmlns="http://servicenet.t-systems.com/medina/xMCF">
                        <data_at_connector>
                            ....
                        </data_at_connector>
                    </MEDINA>
                </appdata>
               ...
            </connection_1d>
            <connection_2d>
               ...
            </connection_2d>
            ...
        </connection_list >
    </connection_group>
    ...
</xmcf>
```

7.6 XML schema definition

The XML schema definition (XSD) of χMCF can be found in computer-interpretable form at the following URL: https://standards.iso.org/iso/pas/8329/ed-1/en/

8 Data common to any connection

8.1 Indices and their properties

χMCF provides several elements which are essentially ordered sets of the same data type (strings, integers, or decimals). More precisely they are like lists or vectors. For example, the `<loc_list/>` for the coordinate list of a seam weld or the `<string_list/>` in the `<custom_attributes/>`. Often the order of the elements in a set is essential. For instance, the coordinates in the `<loc_list/>` for a seam weld define the weld in the space uniquely by their values and their explicit order in the list.

The current XML standard allows that several child elements with an identical name share a common parent. However, it lacks a built-in mechanism to introduce a logical structure (like an order) in an XML-document. χMCF resolves this problem by introducing an index (attribute) in such cases. Indices may play a twofold role: to distinguish from each other and to ensure a unique arrangement in the list. Usually, an index can consist of strictly monotonically increasing natural numbers. In some cases, strictly monotonically increasing real numbers can also be appropriate. For example, the `<loc_list/>` for the coordinate list of a seam weld can be indicated both by real numbers like arc length of the line or any increasing integer series.

Depending on the context, the name of an index (attribute) is `index`, `v` or something else. They are always explicitly stated at the appropriate places in the text.

8.2 Connection referencing

8.2.1 Need for referencing

Any connection should have a way of referring to it, since its shape and dimensions can vary along the design process. Typically, connections are referred to by their assigned ID or label.

8.2.2 Attribute `label`

Any connection should have an attribute called `label`, which identifies it throughout the entire CAx process, maybe even throughout the complete product lifecycle, including manufacturing. It is not necessary that these labels are unique: For instance, if a seam weld is split into different parts at a certain step in the process (if there are interfering holes in the structure, for example), its components shall keep the `label` attribute. A system downstream in the process (detached from any centralized naming authority) may create new connections with all the same label such as “0” or empty string.

The `label` may be composed of digits only, but it should not be confused with other IDs such as a finite element IDs. If desired, finite element IDs would have to be placed within some `<appdata/>` element.

8.2.3 Attribute `ident`

For systems or processes that use integers for referring to connections, the attribute `ident` is provided. In contrast to alphanumeric labels, integers are easy to generate and simple to shift when grouping is needed. This allows for unique identification, detached from a centralized naming authority, in case a connection is split, inserted, or duplicated.

`ident` can be used together with `label` as alternative ways of referring to a connection, bridging the gap between tools that work with integers only and tools that use labels only.

`ident` is a positive integer and unique within the χMCF file.

Example

```
<connection_list>
    <connection_0d label="SPOT_3490" ident="3490">
        <loc> ... </loc>
        <spotweld/>
    </connection_0d>
</connection_list>
```

8.3 Dimensions and coordinates

Connections can have three different dimensions: `<connection_0d/>`, `<connection_1d/>` and `<connection_2d/>`.

Any connection shall have coordinates. How many they are and how they are described depends on the connection’s dimension. Details are described in the following subclauses.

8.4 Attribute `quality_control`

Some connections are more relevant than others, e.g. with respect to crash safety. Therefore, several levels of quality control are well established in manufacturing processes. For this reason, any connection can have an optional attribute `quality_control`. Since there is no general standard for such quality controls, χMCF cannot define a set of possible values for this attribute. Therefore, it must be of type `Alphanumeric`.

8.5 Custom attributes list

It was mentioned in 4.2 that only information relevant to connections should be contained in χMCF. The exceptions `<appdata/>` and `<femdata/>` were introduced in 7.3. These two elements essentially aim at specific needs of application software. The internal structure of `<appdata/>` itself is not standardized, can

be very complex, and depends on the specific software. The content can usually not be interpreted by other software systems.

Frequently, there are situations where a user of χMCF wishes to introduce supplementary information (attributes) to enrich the standard attributes defined by χMCF. In principle, the supplementary information could also be placed in an `<appdata/>` block, but with a substantial drawback, namely, its exchange between different commercial software tools will be difficult in case the tool specific internal structure is not documented.

With `<custom_attributes/>`, χMCF provides an element which is simple in handling and flexible enough to meet many requirements. All descendants of `<custom_attributes/>` are key-value-pairs, following the same pattern *key* ↔ *value(s)*, with supported *value-type* ∈ [*int*, *real*, *string*]^N, where *N* is a positive integer:

(*value-type*) *key* = {*value1, value2, …, valueN*}.

The case *N>1* is reminiscent of the vector or list templates from the STL of C++ (see Reference [8]) and is called "list" in χMCF.

In detail, the individual elements of `<custom_attributes/>` are of one of the following forms:

```
<int key="NameofIntValue"> value </int>
<int_list key="NameofIntListValue">
    <value index="1"> value1 </value>
    …
    <value index="N"> valueN </value>
</int_list>
<real key="NameofRealValue"> value </real>
<real_list key="NameofRealListValue">
    <value index="1"> value1 </value>
    …
    <value index="N"> valueN </value>
</real_list>
<string key="NameofStringValue"> value </string>
<string_list key="NameofStringListValue">
    <value index="1"> value1 </value>
    …
     <value index="N"> valueN </value>
</string_list>.
```

This means that the name of the element specifies the value-type while the value(s) is/are hold in one or several element(s) `<value/>`. A list is signified by the suffix "_list". All elements own the attribute `key`.

Often, the `<custom_attributes/>` element has an owner and is needed for a special purpose. For example, Mr. Brown needs for one and the same joint element an integer valued attribute named "priority" which should take on different values for two different applications "Fatigue" (1) and "Statics" (22). This can be specified in a `<custom_attributes_list/>` as follows:

EXAMPLE

```
<custom_attributes_list>
     <custom_attributes owner="Mr Brown" for="Fatigue">
          <int key="priority"> 1 </int>
     </custom_attributes>
     <custom_attributes owner="Mr Brown" for="Statics">
          <int key="priority"> 22 </int>
     </custom_attributes>
</custom_attributes_list>
```

In the above example, the `owner` is "Mr Brown" in both cases, while the applications can be distinguished by the attributes `for="Fatigue"` and `for="Statics"`, respectively.

The more general case that several `<custom_attributes/>` with different ownerships and for different purposes is considered by the element `<custom_attributes_list/>` with all the `<custom_attributes/>` elements as child-elements. No attributes are associated to the element `<custom_attributes_list/>`.

The existence of a <custom_attributes_list/> inside a connection is optional. There can be at most one element inside each connection.

The <custom_attributes_list/> contains at least one of the following nested elements (see Table 17):

Table 17 — Nested elements of element <custom_attributes_list/>

Nested elements	Multiplicity	Use	Constraint
custom_attributes	1 - *	Required	-

At least one element <custom_attributes/> must be inside a <custom_attributes_list/>.

The XML specification of the <custom_attributes/> element is shown in Table 18:

Table 18 — Attributes of <custom_attributes/> element

Attributes	Type	Value space	Use	Constraints / Remarks
owner	Alphanumeric	Alphanumeric	Required	Non-empty string
for	Alphanumeric	Alphanumeric	Optional	Non-empty string

The pair of attributes owner and for of each <custom_attributes/> element must be unique within each <custom_attributes_list/>.

The <custom_attributes/> element must contain at least one of the following nested elements (see Table 19):

Table 19 — Nested elements of element <custom_attributes/>

Nested elements	Multiplicity	Use	Constraints / Remarks
string	1 - *	optional	At least one of these nested elements is needed.
real	1 - *	optional	
int	1 - *	optional	
string_list	1 - *	optional	
real_list	1 - *	optional	
int_list	1 - *	optional	

The elements <string/>, <real/> and <integer/> must have the following data type assignments for their value, respectively:

— <string/>: alphanumeric value, which is covered by string data type in xsd, which can contain characters, line feeds, carriage returns, and tab characters.

If required to handle special characters such as line feeds or tabs, the normalized string data type should be used in the XSD. The normalized string data type is derived from the string data type. The normalized string data type also contains characters, but the XML processor will remove line feeds, carriage returns, and tab characters.

— <real/>: floating point value, which is covered by decimal data type in xsd, which can contain a numeric value. The maximum number of decimal digits you can specify is 18;

— <integer/>: integer value, which is covered by integer data type in xsd, which can contain a numeric value without a fractional component.

The XML specification of the <string/> element is shown in Table 20:

Table 20 — Attributes of <string/> element

Attributes	Type	Value space	Use	Constraints / Remarks
key	Alphanumeric	Alphanumeric	Required	Non-empty string

The XML specification of the <real/> element is shown in Table 21:

Table 21 — Attributes of <real/> element

Attributes	Type	Value space	Use	Constraints / Remarks
key	Alphanumeric	Alphanumeric	Required	Non-empty string

The XML specification of the <integer/> element is shown in Table 22:

Table 22 — Attributes of <integer/> element

Attributes	Type	Value space	Use	Constraints / Remarks
Key	Alphanumeric	Alphanumeric	Required	Non-empty string

The XML specification of the <string_list/> element is shown in Table 23:

Table 23 — Attributes of <string_list/> element

Attributes	Type	Value space	Use	Constraints / Remarks
key	Alphanumeric	Alphanumeric	Required	Non-empty string

<string_list/> has the nested element (see Table 24):

Table 24 — Nested elements of <string_list/> element

Nested elements	Type	Multiplicity	Use	Constraints / Remarks
value	**Alphanumeric**	**1 - ***	**required**	**-**

Where <value/> within <string_list/> is specified as (see Table 25):

Table 25 — Attributes of <value/> element inside <string_list/>

Attributes	Type	Value space	Use	Constraints / Remarks
index	Integer	>0	Required	unique within the parent element

The XML specification of the <real_list/> element is shown in Table 26:

Table 26 — Attributes of <real_list/> element

Attributes	Type	Value space	Use	Constraints / Remarks
key	Alphanumeric	Alphanumeric	Required	Non-empty string

<real_list/> has the nested element as (see Table 27):

Table 27 — Nested element of <real_list/> element

Nested elements	Type	Multiplicity	Use	Constraints / Remarks
value	**Floating point**	**1 - ***	**Required**	**-**

Where <value/> within <real_list/> is specified as (see Table 28):

Table 28 — Attributes of <value/> element inside <real_list/>

Attributes	Type	Value space	Use	Constraints / Remarks
index	integer	>0	Required	unique within the parent element

The XML specification of the <int_list/> element is shown in Table 29:

Table 29 — Attributes of `<int_list/>` element

Attributes	Type	Value space	Use	Constraints / Remarks
key	Alphanumeric	Alphanumeric	Required	Non-empty string

`<int_list/>` has the nested element (see Table 30):

Table 30 — Nested elements of `<int_list/>` element

Nested elements	Type	Multiplicity	Use	Constraints / Remarks
value	**integer**	**1 - ***	**Required**	**-**

Where `<value/>` within `<int_list/>` is specified as (see Table 31):

Table 31 — Attributes of `<value/>` element inside `<real_list/>`

Attributes	Type	Value space	Use	Constraints / Remarks
index	integer	>0	Required	unique within the parent element

Remarks:

a) Values of `key` must be unique within their common parent element,

b) The order of the values in the corresponding list is identified by the numerical value of their attribute `index`, in ascending order. Therefore, indices must be unique within one list,

c) In case of strings, the whitespaces deserve extra mention: To avoid mistakes, whitespaces are not to be used at beginning and end of a string.

EXAMPLE

```
<custom_attributes_list>
      <custom_attributes owner="DepartmentA" for="Fatigue">
            <int key="priority"> 1 </int>
            <string key="used S-N curve">Steel_225_ISO</string>
            <real key="fatigue_limit"> 223.1 </real>
      </custom_attributes>
      <custom_attributes owner="DepartmentA" for="Statics">
            <int key="priority"> 2 </int>
      </custom_attributes>
      <custom_attributes owner="DepartmentB">
            <string key="priority">high</string>
            <real_list key="direction vector">
                  <value index="1">10.3 </value>
                  <value index="2"> -2.1</value>
                  <value index="3">-1.5</value>
            </real_list>
            <string_list key="verifiedby" >
                  <value index="1">john</value>
                  <value index="2">Smith</value>
            </string_list>
      </custom_attributes>
</custom_attributes_list>
```

8.6 Distinction between `<custom_attributes/>` and `<appdata/>`

8.6.1 General

At first glance, `<custom_attributes/>` and `<appdata/>` seem to address similar purpose or even to be redundant. This is misleading, as the following subclauses show.

8.6.2 Needs of different process roles, addressed by <custom_attributes/> and <appdata/>

In the context of χMCF, at least two different roles can be distinguished:

— the programmer of an application, and

— the engineer using this application.

The programmer needs to store extra data that are specific to the application.

The engineer needs to store additional data specific to the process in which the connections are involved.

As its name suggests, <appdata/> is used to store application-specific data, whose structure and purpose is known only by the application itself, respectively applies to this application alone. The software vendor can choose to standardize and publish the format of this data in order to allow other applications to port data to it or may choose to use <appdata/> as a private storage of internal state.

<custom_attributes/> represent OEM- or process-specific data, whose purpose is known by the engineers, but may not be known by the application.

Engineers choose which attributes they need to store and designate the corresponding data in <custom_attributes/>.

Applications store auxiliary data in <appdata/>. These data may be data that engineers do not need to know. <appdata/> may include information about the internal state of the application specific data model.

Engineers know the purpose and representation of <custom_attributes/>. The software may not know what each custom attribute represents, but it shall nevertheless be able to transport these data unchanged, or to offer a (generic) graphical user interface (GUI) for accessing it.

8.6.3 Needs of different applications, addressed by <custom_attributes/> and <appdata/>

<appdata/> may be used as means of intercommunication between different applications. In this case, the format of <appdata/> needs to be documented and published by the <appdata/> owner. However, it is not mandatory that information stored in <appdata/> is handled, maintained, or processed by third-party software. Therefore, <appdata/> should be considered as data that can be disregarded or thrown away by a third-party party software. Therefore, applications shall not rely on preservation of <appdata/>. Data corruption or crash must be avoided if data from <appdata/> gets lost.

The internal structure of <custom_attributes_list/> is completely standardized, whereas the internal structure of <appdata/> is arbitrary and can for optionally be described by a software-specific XML schema. Therefore, <custom_attributes_list/> cannot be used as flexible as <appdata/>, but its content is easier to be preserved across system boundaries.

8.6.4 Different levels of <custom_attributes/> and <appdata/> within χMCF data model

The elements <appdata/> and <custom_attributes_list/> have been introduced in 7.3.2 and 8.5, respectively. In this subclause, they are compared regarding their arrangement in the XML tree.

<appdata/> may be used either on different levels of a χMCF file:

— on root level (directly within the <xmcf/> element), or

— within any single connector (elements <connection_0d/>, <connection_1d/> and <connection_2d/>)

— or both.

In contrast to this, the element <custom_attributes_list/> can only be used within any single connector, but not at root level. This is justified below.

In the usual scenario where multiple χMCF files, each containing connections of subsystems, are to be read and combined in an application, the application must deal with possible conflicts between data at root level (which applies to many connectors) and data at connector level (which applies to a single connector, only).

Each application should be able to resolve conflicts in its own `<appdata/>`, as the semantics of the data is known to the application.

On the other hand, the purpose of a `<custom_attributes/>` element assumed at root level would not be known to a random application. Conflicts with `<custom_attributes/>` elements at connector level would not even be identifiable with certainty. The application would therefore have to pass on the task of resolving `<custom_attributes/>` conflicts to the user. This is undesirable.

Because `<custom_attributes/>` may only be within a single connector, no conflicts can arise. Any application should be able to handle `<custom_attributes/>`, even if it does not understand the semantics. The reason for this is that the syntax is standardised and that connectors are self-contained objects with a clearly defined role and limited scope.

9 0D connections

9.1 Generic definitions

9.1.1 Identification

9.1.1.1 General

The possible XML attributes of the `<connection_0d/>` element, which describes a point connection (0D connection), are specified in Table 32:

Table 32 — Attributes of element `<connection_0d/>`

Attributes	Type	Use	Constraint
label	Alphanumeric	Optional	-
ident	Integer	Optional	positive, unique within a χMCF file
quality_control	Alphanumeric	Optional	See 8.4.

9.1.1.2 Attribute “label”

The label defines the human readable identification of a connection. It can contain a description of the connection or simply an index as an integer.

EXAMPLE 1 Minimum definition of a 0d connection without label

```
<connection_list>
    <connection_0d>
        <loc>
            ...
        </loc>
        <spotweld>
            ...
        </spotweld>
    </connection_0d>
</connection_list>
```

EXAMPLE 2 Definition of a 0d connection with label

```
<connection_list>
    <connection_0d label="SPOT_3490">
        <loc>
            ...
        </loc>
        <spotweld>
```

```
            ...
        </spotweld>
    </connection_0d>
</connection_list>
```

9.1.1.3 Attribute "ident"

The attribute ident provides an alternative identification to the connection. The value of ident is a positive integer and unique within the χMCF file.

EXAMPLE

```
<connection_list>
    <connection_0d ident="3490">
        <loc>
            ...
        </loc>
        <spotweld>
            ...
        </spotweld>
    </connection_0d>
</connection_list>
```

9.1.2 Location

The definition of the connection location of a point connection is described by the element <loc/>. This element is nested below the parent element <connection_0d/>. It contains three values specifying the x, y, and z coordinates of the location as text content (Table 33).

Table 33 — Text values of element <loc/>

Text	Type	Value space	Use	Constraint
x	Floating point	(−∞, +∞)	Required	-
y	Floating point	(−∞, +∞)	Required	-
z	Floating point	(−∞, +∞)	Required	-

EXAMPLE

```
<connection_0d>
    <loc> 2581.21 -708.408 31.6532 </loc>
    ...
</connection_0d>
```

9.1.3 Direction

The definition of connection directions, where applicable, is described by the elements <normal_direction/> and <tangential_direction/>. They both specify a direction vector. Lengths of both vectors are not relevant but must be > 0.

Their XML syntax is identical. Their names describe their semantics:

— Element <normal_direction/> denotes a direction of a local z axis.

— Element <tangential_direction/> denotes the direction of an axis tangential to the (base) part surface next to the point given in <loc/>, giving locale x axis. Its orthogonalization relative to <normal_direction/> is not allowed to vanish, that means both vectors are not allowed to be collinear.

If both elements are given, a right-handed coordinate system is uniquely defined:

— Origin is in <loc/>.

— z-axis is in direction of <normal_direction/>.

— x-axis is the orthogonalization of <tangential_direction/> relative to <normal_direction/>.

— y-axis is given by cross product z-axis × x-axis.

The XML specification of the `<normal_direction/>` and `<tangential_direction/>` elements is shown in Table 34:

Table 34 — Attributes of elements `<normal_direction/>` & `<tangential_direction/>`

Attributes	Type	Value space	Use	Constraint
x	Floating point	(−∞, +∞)	Required	-
y	Floating point	(−∞, +∞)	Required	-
z	Floating point	(−∞, +∞)	Required	-

Both elements do not allow for any nested elements.

Unless otherwise stated, direction elements are optional. However, if omitted, importing systems may use a geometric search for determining `<normal_direction/>`, using a spherical characteristic, which may not be very reliable. In this situation, `<tangential_direction/>` can only be guessed, implying a random orientation of the connection (e.g. a robscan) in receiving system. Therefore, it is recommended for a receiving system to issue a warning, at least.

EXAMPLE

```
<normal_direction x="0.0" y="0.0" z="-1.0" />
<tangential_direction x="70.7" y="70.7" z="0.0" />
```

9.1.4 Type specification

Each connection should be assigned a type during its life cycle. The XML definitions of all 0D connections contain the following elements (see Table 35):

Table 35 — Nested elements of element `<connection_0d/>`

Nested elements	Multiplicity	Use	Constraint
clinch	1	Optional	-
clip	1	Optional	-
heat_stake	1	Optional	-
nail	1	Optional	-
gumdrop	1	Optional	-
rivet	1	Optional	-
robscan	1	Optional	-
rotation_joint	1	Optional	-
spotweld	1	Optional	-
threaded_connection	1	Optional	-
contact_list	1	Optional	See 7.4.3.6.
stacking	1	Optional	See 7.4.2.4

Up to one of the type elements (`clinch`, `clip`, `heat_stake`, `gumdrop`, `nail`, `rivet`, `robscan`, `spotweld`, or `threaded_connection`) may exist in `<connection_0d/>`. There is no default type.

9.2 Spot welds

9.2.1 General

A spot weld is denoted by an element `<spotweld/>`. This element is described completely by its attributes and nested elements (see Table 36).

Table 36 — Nested elements of `<connection_0d/>` for `<spotweld/>`

Nested elements	Multiplicity	Use	Constraints / Remarks
spotweld	1	Optional	-
loc	1	Required	-
appdata	1	Optional	See 7.3.2.
femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

The XML specification of the `<spotweld/>` with element `diameter` is shown in Table 37:

Table 37 — Attributes of element `<spotweld/>`

Attributes	Type	Value space	Use	Constraint
diameter	Floating point	> 0,0	Optional	-
technology	Selection	resistance, laser, projection, friction	Optional	-

9.2.2 Attribute `diameter`

The diameter of a spotweld is specified by the attribute `diameter` for the child element of `<connection_0d/>`.

9.2.3 Attribute `technology`

The technology used to weld the connection can be specified for each of the spot welds of a connection separately.

This technology can be one of:

— Resistance welding,

— Laser welding,

— Projection welding,

— Friction welding.

The element `<spotweld/>` allows for following nested elements (see Table 38):

Table 38 — Nested elements of element `<spotweld/>`

Nested elements	Multiplicity	Use	Constraint
normal_direction	1	Optional	-
tangential_direction	1	Optional	-

EXAMPLE

```
<connection_0d label="SPOT_Left_Gh_2123921">
    <spotweld diameter="5.0"/>
    <loc> 1645.83 821.145 616.585 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

9.3 Robscans

A Robscan is a pattern of arbitrary shape, drawn onto the flange partners by a laser optic. Such a shape has a length and width significantly larger than the diameter of the laser focus. The laser beam defines a local z-axis and is assumed to be perpendicular to the flange partners. However, the pattern can be rotated around this z-axis, and it can be mirrored at its x-axis. This is depicted in Figure 8:

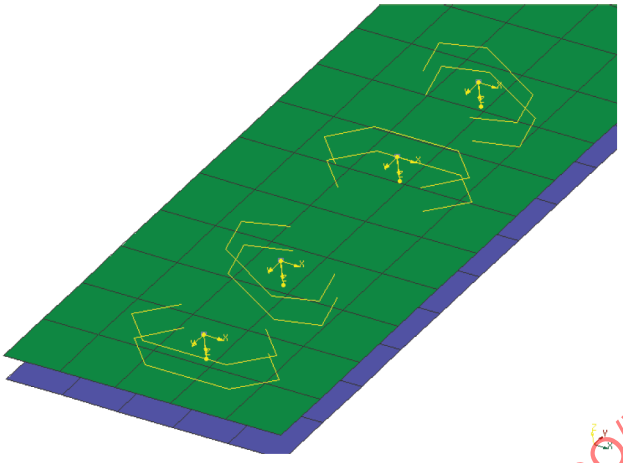


Figure 8 — Robscans with different rotation angles, two of them mirrored

The pattern of the bottom left Robscan is oriented with no rotation and no mirroring with respect to its own coordinate system (yellow). The next instance has 30° rotation. The two Robscans, top right in the figure, have a mirrored pattern; the uppermost having again 30° rotation.

There is a continuum of patterns for Robscans. Each one which shall be used at an assembly line needs to be verified (by simulation plus test) in advance, which is expensive. Some implications are:

— companies regard this information to be their own intellectual property,

— a pattern shall not simply be stretched etc. It would need a new validation,

— validated Robscan patterns are usually not part of distributions of FE processors.

However, subcontractors possibly need to know the position and bounding box of the Robscan. Therefore, χMCF definition shall contain some "abstract" data. FE processors may address the danger of inconsistency by taking both levels of information from the same configuration file. Therefore, it is at the responsibility of the companies' admins to have consistent data in that file.

Since the exact shape of the Robscan pattern is third-party intellectual property, it cannot be part of the χMCF definition. It is referred to by just a string attribute "pattern". Possible values of attribute "pattern" are not the subject of this document: In general, they are very original equipment manufacturer (OEM) specific. However, to provide a minimum amount of information, width and length of the pattern are given by attributes `pattern_width` and `pattern_length`.

A Robscan is denoted by an element `<robscan/>`. This element is described completely by its attributes and nested elements, see Table 39, Table 40, and Table 41.

Table 39 — Nested elements of `<connection_0d/>` for `<robscan/>`

Nested elements	Multiplicity	Use	Constraints / Remarks
robscan	1	Optional	-
loc	1	Required	-
appdata	1	Optional	See 7.3.2.
femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

The XML specification of the `<robscan/>` element is shown in the following Table 40:

Table 40 — Attributes of element `<robscan/>`

Attributes	Type	Value space	Use	Constraints / Remarks
base	Integer	> 0	Optional	-
pattern	Alphanumeric	Alphanumeric	Optional	Non-empty, if present.
gap	Floating point	>= 0,0	Optional	-
width	Floating point	> 0,0	Optional	-
pattern_width	Floating point	> 0,0	Optional	-
pattern_length	Floating point	> 0,0	Optional	-
mirrored	Boolean	"false" (default), "true"	Optional	-
orientation_angle	Floating point	[-180°, 180°]	Optional	According to the unit of angles, defined in element `<units/>`.
filler_material	Alphanumeric	Alphanumeric	Optional	-

All attributes of element `<robscan/>` are optional for import to CAD or CAE processors. However, specific FE solvers may declare some of them to be mandatory.

General defaults are:

— `false` – for Boolean values,

— `0` – for numeric values,

— `""` – for strings.

However, these defaults are not always useful for CAE:

— `gap`: this defines the gap between both flange partners (perpendicular to the surface);

— `width`: this is the width of the laser beam;

— Width and length of the pattern are given by attributes `pattern_width` and `pattern_length`;

— `mirrored`: this denotes, whether the pattern has to be mirrored along its length-axis x, i.e. local y coordinate has to be inverted;

— `orientation_angle`: this defines a rotation around z axis, following right-hands-rule. Angle is measured in the unit of angles, defined in element `<units/>`, within range [–180°, 180°]. –180° and +180° degree are regarded to be identical (applies also to equivalent ranges in another unit).

Both parameters, `mirrored` and `orientation_angle` address numerical optimization: An angle and a boolean allow to vary the Robscan placement easier than to calculate completely new orientation vectors.

Table 41 — Nested elements of element `<robscan/>`

Nested elements	Multiplicity	Use	Constraint
normal_direction	1	Optional	-
tangential_direction	1	Optional	-

Additional explanations for the directions are:

— Element `<normal_direction/>` denotes direction of laser beam, giving local z axis;

— Element `<tangential_direction/>` denotes laser moving direction, giving local x axis.

`<normal_direction/>` and `<tangential_direction/>` elements are described in 9.1.3.

EXAMPLE

```
  <connection_0d label="RSC_1272360">
      <loc> 507 1 0.8 </loc>
      <robscan base="1" pattern="KL_ST" gap="0.15" width="0.4" mirrored="false" pattern_
width="5" pattern_length="12" orientation_angle="0">
          <normal_direction x="0" y="0" z="-1"/> <!-- locale z axis -->
          <tangential_direction x="1" y="0" z="0"/> <!-- locale x axis -->
      </robscan>
      <appdata>
          ...
      </appdata>
  </connection_0d>
```

9.4 Rivets

9.4.1 General

There are many different types of rivets. If at some state of the model the specific type of rivet (e.g. blind rivet, self-piercing rivet) is not known, then a generic rivet element should be used to capture just the necessary information, like direction, length, and diameter.

A rivet is denoted by an element `<rivet/>`. This element is described completely by its attributes and nested elements, see Table 42 and Table 43.

Table 42 — Nested elements of `<connection_0d/>` **for** `<rivet/>`

Nested elements	Multiplicity	Use	Constraints / Remarks
rivet	1	Optional	-
loc	1	Required	-
appdata	1	Optional	See 7.3.2.
femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

The XML specification of the `<rivet/>` element is shown in Table 43 and rivet head types are shown in Figure 9:

Table 43 — Attributes of element `<rivet/>`

Attributes	Type	Value space	Use	Constraints / Remarks
hardness	Floating point	> 0,0	Optional	-
shaft_diameter	Floating point	> 0,0	Optional	-
length	Floating point	> 0,0	Optional	-
head_diameter	Floating point	> 0,0	Optional	-
head_height	Floating point	≥ 0,0	Optional	If at least one of them is specified *head_height + sink_size > 0* is required.
head_type	Alphanumeric	Alphanumeric	Optional	-
sink_size	Floating point	≥ 0,0	Optional	-
strength_property_class	Alphanumeric	Alphanumeric	Optional	-
part_code	Alphanumeric	Alphanumeric	Optional	-

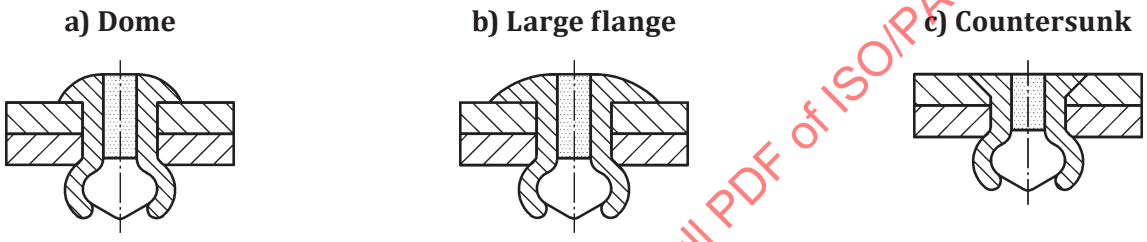


Figure 9 — Rivet head types (dome, large flange, countersunk)

The following list explains the attributes:

— `hardness`: Vickers hardness HV (see Reference [9]) of the rivet material. (Attribute hardness was moved from element `<self_piercing/>` to element `<rivet/>` with χMCF version 3.1.);

— `shaft_diameter`: the diameter of the shaft of the (unmounted) rivet;

— `length`: the overall length of the (unmounted) rivet itself;

— `head_diameter`: the diameter of the head of the (unmounted) rivet;

— `head_height`: the height of the head;

— `head_type`: the description of head type ("dome", "countersunk" or "large_flange");

— `sink_size`: the size of the head that is sunk;

— `strength_property_class`: Strength according to ISO, EN, BSW, DIN, SAE etc., e.g. Reference [10].

— `part_code`: the part code of the rivet, as used e.g. in a PDM system. It can be convenient to use the rivet norm (according to e.g. ISO, EN, BSW, DIN) as part code.

If possible, a rivet should know the direction of fixation, therefore, possess a nested element `<normal_direction/>`. However, this is not mandatory so that incomplete data can also be imported. The direction sense of `<normal_direction/>` (see 9.1.3) is from rivet head to foot.

A `<tangential_direction/>` can be provided for rivets that are not axis-symmetric and require a spatial orientation.

A `<rivet/>` is always placed into holes drilled before, whereas its subtype `<self_piercing/>` creates its own hole during placement.

Specific subtypes of rivets are defined by adding nested elements, listed in following Table 44:

Table 44 — Nested elements of element `<rivet/>`

Nested elements	Multiplicity	Use	Constraint / Remarks
normal_direction	1	Optional	-
tangential_direction	1	Optional	-
blind self_piercing solid swop clinch_rivet_stud	1	Optional	Maximum one of the listed elements.

The subtypes are described in detail in the 9.4.2 to 9.4.6.

EXAMPLE 1 Example for a (axisymmetric) rivet connection that uses only the <normal_direction/>

```
<connection_0d label="RVT_2123921">
    ...
    <rivet shaft_diameter="5.0" head_diameter="8" length="3.5">
        <normal_direction x="0" y="0" z="3"/>
    </rivet>
    <loc> 1645.83 821.145 616.585 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

EXAMPLE 2 Example for a rivet connection that requires also the `<tangential_direction/>`

```
<connection_0d label="RVT_2123922">
    ...
    <rivet shaft_diameter="5.0" head_diameter="8" length="3.5">
        <normal_direction x="0" y="0" z="3"/>
        <tangential_direction x="3" y="0" z="0"/>
    </rivet>
    <loc> 1645.83 -821.145 616.585 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

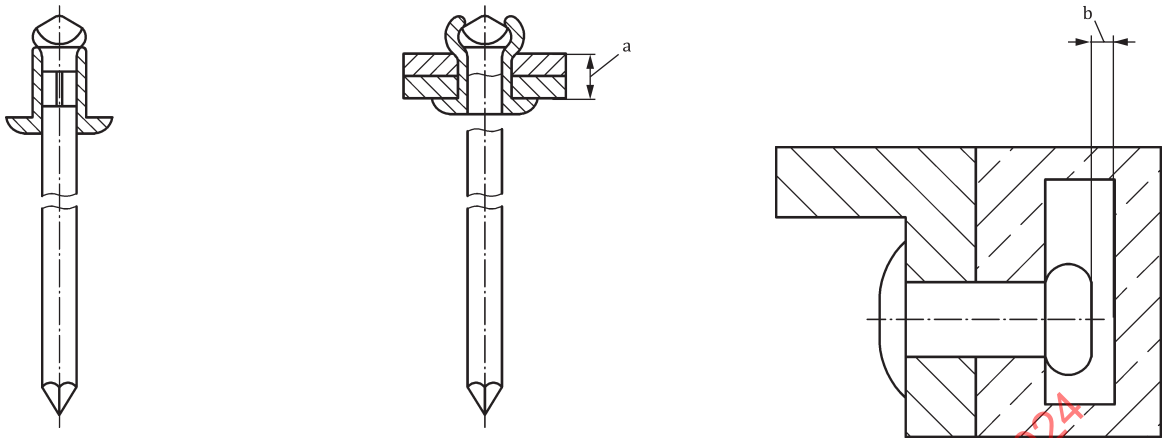
9.4.2 Blind rivets

Blind rivets are one-sided rivets that require a pre-drilled hole. Blind rivets form their shape when the mandrel is pulled out from the rivet body. This action securely clamps the sheets together. A blind rivet is denoted by a nested element `<blind/>` within `<rivet/>`. This element is described completely by its attributes and those of `<rivet/>`.

The XML specification of the `<blind/>` element is shown in Table 45:

Table 45 — Attributes of element `<blind/>`

Attributes	Type	Value space	Use	Constraint / Remarks
min_grip	Floating point	> 0,0	Optional	-
max_grip	Floating point	> 0,0	Optional	greater equal to min_grip
clearance	Floating point	> 0,0	Optional	-
material	Alphanumeric	Alphanumeric	Optional	material of the rivet body



a) Before riveting **b) After riveting** **c) Application example**

a Grip.

b Clearance (blind side).

Figure 10 — Blind rivet — Key attributes

Figure 10 describes what the attributes of `<rivet/>` and `<blind/>` correspond to:

— `min_grip`, `max_grip`: These two attributes collectively describe the effective grip range of the rivet. A blind rivet is engineered so that it can be used for a specific range of material thickness for which it provides proper joining between connected parts. This can be called as the blind rivet's grip range.

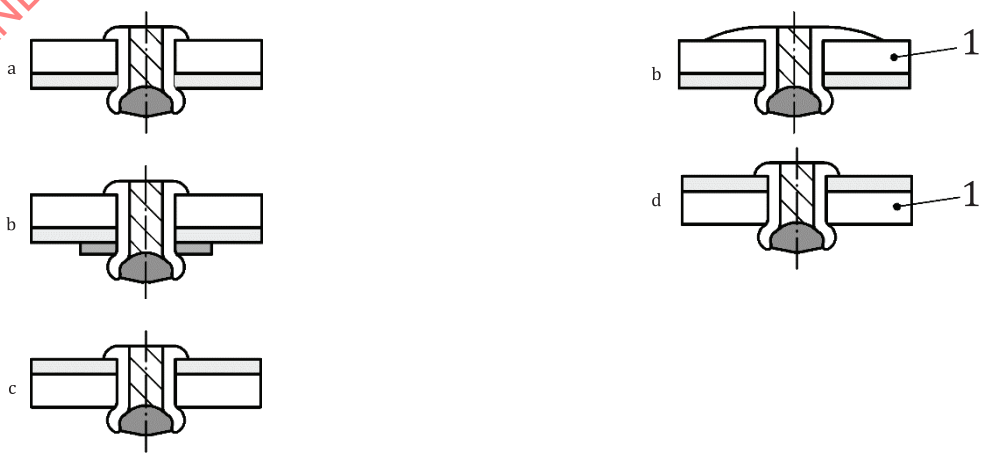
— `clearance`: The blind rivet needs some clearance on the blind side, which is the side of mandrel head, when inserted into the holes, before it is applied.

— `material`: This attribute defines the applied material of the blind rivet body. Generally, the applied rivet should be used with connected parts so that the connector rivet element has the same physical and mechanical properties as the components to be joined. Usual materials: Steel, Stainless Steel, Nickel Copper Alloy (Monel), Copper and several grades of Aluminium.

NOTE If material thickness changes in connected parts, this can lead to other size of blind rivet as joining element.

If a blind rivet is applied to join two components which have different mechanical properties, e.g. one of them is thinner or softer than the other, then the `<normal_direction/>` element will become more important to show the proper setting direction of the rivet as seen in Figure 11.

a) Thin / Thick assembly **b) Soft / Hard assembly**



Key

1 soft part

[a] Satisfactory.

[b] Good.

[c] Better.

[d] Poor.

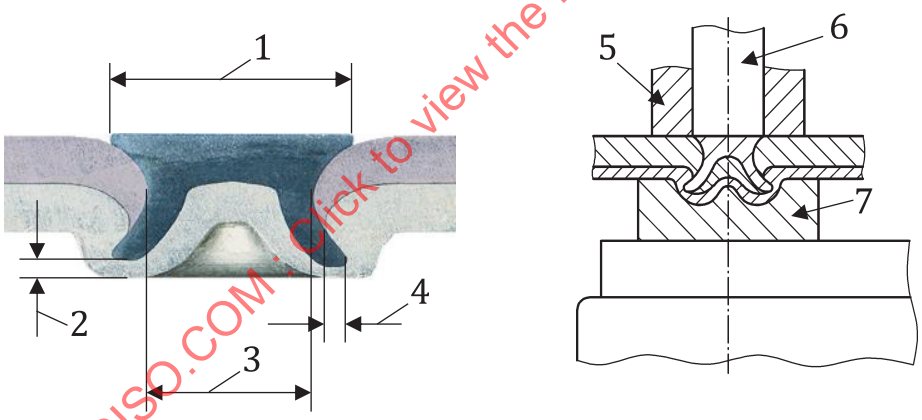
Figure 11 — Assembly recommendations for blind rivets

EXAMPLE

```
<connection_0d label="RVT_2123921">
    <rivet shaft_diameter="3.35" head_diameter="5.5" head_type="dome" length="4">
           <blind min_grip="3" max_grip="3.2" clearance="4.5" material="Steel"/>
           <normal_direction x="0.0" y="1.5" z="3.0"/>
    </rivet>
    <loc> 1645.83 821.145 616.585 </loc>
    <appdata>
           ...
    </appdata>
</connection_0d>
```

9.4.3 Self-piercing rivets

A self-piercing rivet is a special kind of rivet which does not need a pre-drilled hole. Originally a hollow cylinder with a cap on one end, it deforms together with the material it is pushed into as shown below in Figure 12:



Key

1 rivet head diameter

2 bottom thickness

3 rivet diameter

4 undercut

5 blank holder

6 punch

7 die

Figure 12 — Cross-section of a self-piercing rivet and riveting device

A wide range of such rivets is available. They can be used with different rivet dies to optimize the riveting process. Such combinations must be chosen in accordance with the materials of the flange partners. Details are third party intellectual property and therefore cannot be part of the χMCF definition. However, χMCF offers alphanumeric attributes for rivet and die parameters. Possible values of these attributes are

not subject of this document: In general, they are very OEM specific. To provide a minimum amount of information, some general geometric information is given by related numerical attributes.

A self-piercing rivet is denoted by a nested element `<self_piercing/>` within `<rivet/>`. This element is described completely by its attributes and those of `<rivet/>`. In especially, attributes `length`, `head_diameter` and `shaft_diameter` are inherited from `<rivet/>`.

The XML specification of the `<self_piercing/>` element is shown in Table 46:

Table 46 — Attributes of element `<self_piercing/>`

Attributes	Type	Value space	Use	Constraint
head_label	Alphanumeric	Alphanumeric	Optional	-
shaft_label	Alphanumeric	Alphanumeric	Optional	-
die_label	Alphanumeric	Alphanumeric	Optional	-
die_diameter	Floating point	> 0,0	Optional	-
die_depth	Floating point	> 0,0	Optional	-

All attributes of this connection are optional for import to CAD or CAE processors. However, specific FE solvers may declare some of them to be mandatory.

The `head`, `shaft` and `die` labels are very OEM specific. However, to provide a minimum amount of information, diameters of them plus depth of die are given.

The attribute `die_label` can be used to refer to a catalogue entry. In this situation, `die_diameter` and `die_depth` can be omitted in χMCF file, if their values are given in the catalogue.

One level higher, the entire rivet can refer via an attribute to an item that is found in an OEM Specific PDM system. In this case, subtype definition is used from catalogue, too, if present. To maintain consistency in such cases, the `<rivet/>` in χMCF file is not allowed to specify another subtype than the referred item from the PDM system.

General defaults for attributes are: 0 for numeric values, "" for strings. However, these defaults are not always useful for CAE.

EXAMPLE

```
<connection_0d label="RVT_2123921">
    <rivet shaft_diameter="3.35" head_diameter="5.5" length="4" hardness="410">
           <normal_direction x="0" y="0" z="3"/>
           <self_piercing head_label="N000000002651" shaft_label="C"
           die_depth="2.5" die_label="DZ11x2,5-0,50" die_diameter="11" />
    </rivet>
    <loc> 1645.83 821.145 616.585 </loc>
    <appdata>
           ...
    </appdata>
</connection_0d>
```

9.4.4 Solid rivets

Solid rivets require a pre-drilled hole. They can be found in many similar forms, with a cap on one end. The other end deforms when it is pushed from the other side.

Shafts of solid rivets are typically solid, but can be designed differently in detail, as shown in Figure 13:

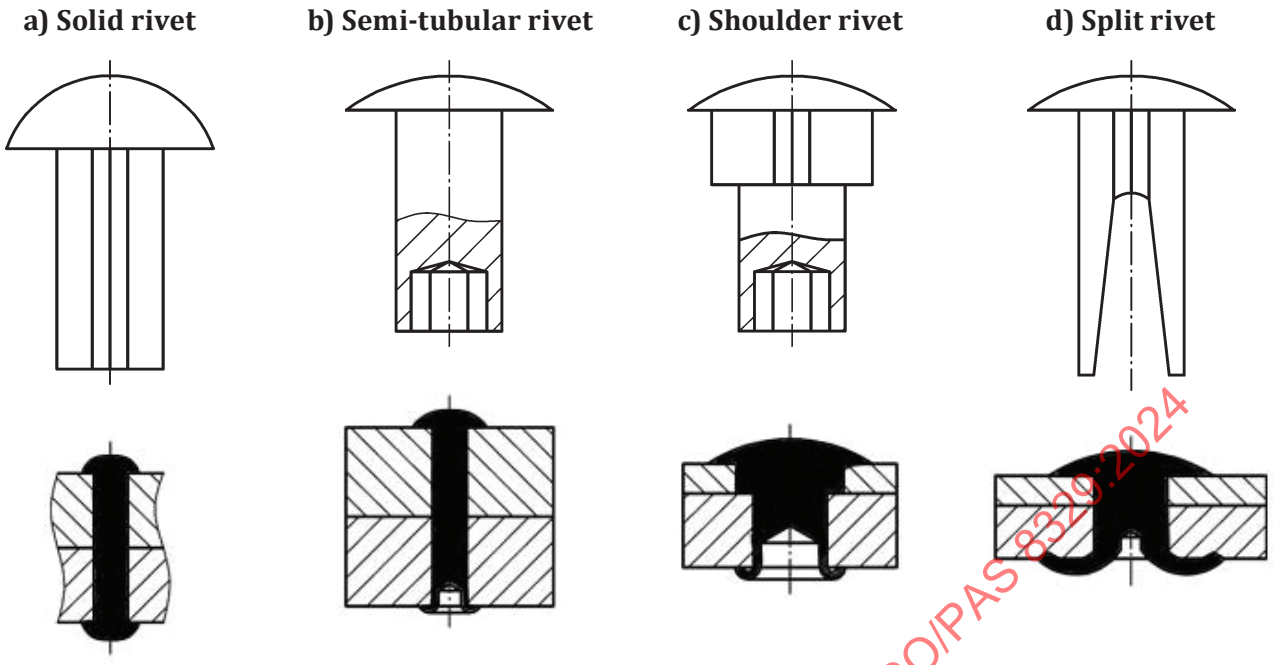
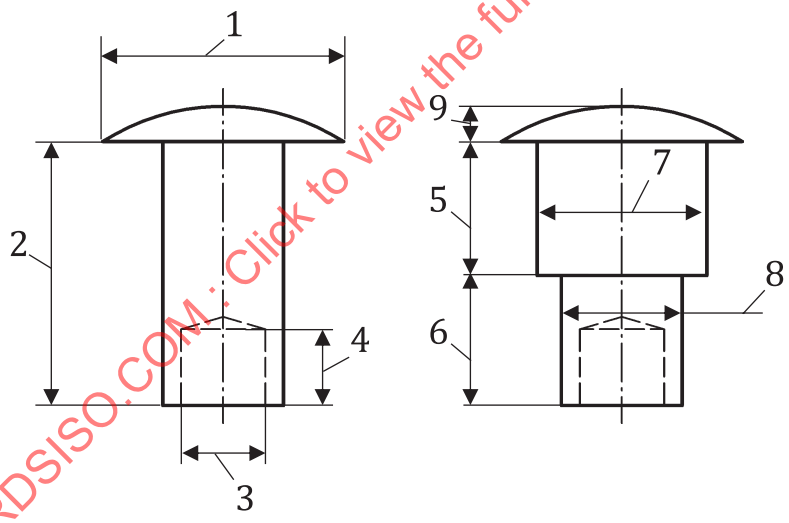


Figure 13 — Pictures of characteristic rivet types before and after mounting

Key dimensions of all these rivets generalize into the diagram shown in Figure 14:



Key

1 head diameter
2 length
3 hole diameter
4 hole depth
5 shoulder length
6 tenon length
7 shoulder diameter
8 tenon diameter
9 head heigt

Figure 14 — Key dimensions of solid rivets

A solid rivet is denoted by a nested element `<solid/>` within `<rivet/>`. This element is described completely by its attributes and those of `<rivet/>`.

The XML specification of the `<solid/>` is shown in Table 47:

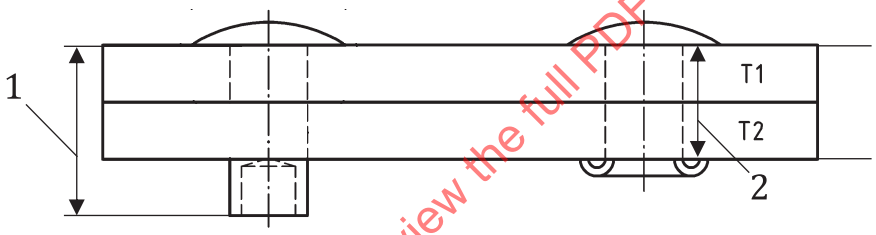
Table 47 — Attributes of element `<solid/>`

Attributes	Type	Value space	Use	Constraint
min_grip	Floating point	> 0,0	Optional	-
max_grip	Floating point	> 0,0	Optional	max_grip ≥ min_grip
hole_diameter	Floating point	> 0,0	Optional	-
hole_depth	Floating point	> 0,0	Optional	-
shoulder_diameter	Floating point	> 0,0	Optional	-
shoulder_length	Floating point	> 0,0	Optional	-
tennon_diameter	Floating point	> 0,0	Optional	-
tennon_length	Floating point	> 0,0	Optional	-

The following recommendations apply:

a) `hole_diameter` is defined together with `hole_depth` and vice versa.

b) `tennon_diameter` exist only if `shoulder_diameter` is defined and vice versa.



Key

1 rivet length

2 grip

Figure 15 — Relation of working thickness (T1+T2) to max and min values of grip

Figure 14 and Figure 15 describe what the attributes of `<rivet/>` and `<solid/>` correspond to:

— `min_grip, max_grip`: These two attributes collectively describe the effective grip range.

— `hole_diameter`: This is the diameter of the hole of the tube. This value is provided in a supplier standard normally.

— `hole_depth`: This is a measure of the hole of the tube. There is no exact relation between `hole_depth` and grip range. Dependent of the supplier, it can be a length calculation that can result in an advised clinch allowance based on the work thickness calculated by the sum of the thicknesses of connected parts.

— `shoulder_diameter, shoulder_length`: The rivet's shoulder sizes. Note that shoulder length is typically measured next under the head.

— `tennon_diameter, tennon_length`: These attributes describe the secondary smaller shoulder sizes. A `tennon_diameter` should not exist without a primary `shoulder_diameter`.

If a `head_height` exists, `sink_size` will be 0, and vice versa (see explanation after Figure 9), but there is no formal constraint in χMCF.

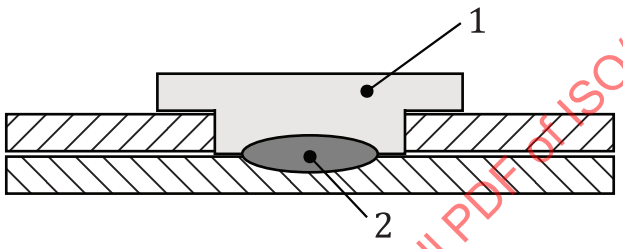
EXAMPLE

```
  <connection_0d label="RVT_2123921">
      <loc> 1645.83 821.145 616.585 </loc>
      <rivet shaft_diameter="3.35" head_diameter="5.5" head_height="0.4" length="4">
            <solid min_grip="3" max_grip="3.2" hole_depth="0.8" shoulder_diameter="3.8"
shoulder_length="1.2"/>
            <normal_direction x="0" y="1.5" z="3"/>
      </rivet>
      <appdata>
            ...
      </appdata>
  </connection_0d>
```

9.4.5 Swop rivets

The sheet weld opposed plug (SWOP) method is used to connect parts with spot welds in cases where one component material is not suitable to create any alloy with the other part's material. This is the case, for example, aluminium and steel parts are to be connected. An example of such a method is given in the European patent EP 0967044 A2[11].

Figure 16 shows a cross-section of a SWOP:



Key

1 insert

2 spotweld

Figure 16 — Cross-section of a SWOP rivet

The common technological challenge addressed by SWOP methods is to join a shell component made of a material that cannot be electrically welded (attached part) to a weldable shell component (base part).

For this purpose, a hole in the attached part is filled with a button-shaped insert made of weldable material. This insert is welded to the base part and ensures the connection due to its geometric button shape.

Based on the description above a wide range of insert shapes can be imagined. Therefore, the details of these shapes cannot be part of the χMCF specification. A shape is referred by a string attribute `insert_shape`. The possible values of this attribute are not subject of this document. In general, they are very OEM specific. However, to provide a minimum amount of information, some general geometric data are given by the attributes introduced below.

A SWOP rivet is denoted by a nested element `<swop/>` within `<rivet/>`. This element is described completely by its attributes and parent element attributes within `<rivet/>`. In particular, the attributes `shaft_diameter`, `sink_size`, `length`, `head_diameter` and `head_height` are inherited from the `<rivet/>` element. Other rivet parameters (such as `length` or `shaft_diameter`) may be treated as meaningless.

The XML specification of the `<swop/>` element is shown in Table 48:

Table 48 — Attributes of element `<swop/>`

Attributes	Type	Value space	Use	Constraint
insert_shape	Alphanumeric	Alphanumeric	Optional	-
insert_height	Floating point	> 0,0	Optional	-
spotweld_diameter	Floating point	> 0,0	Optional	-
spotweld_technology	Selection	resistance laser projection friction	Optional	-

All attributes of this connection type are optional for importing it into CAD or CAE application. However, some FE pre-processors may declare some of them to be mandatory.

These attributes have the following semantics:

— `insert_shape`: Identification of the applied insert shape. In the illustrated example, the hole appears circular, but it may have a polygonal shape in order to prevent relative rotation of the two sheets in case they were connected by a single framing spot, only.

— `insert_height`: Height of the (unmounted) insert.

— `spotweld_diameter`: Diameter of the spot weld, see 9.2 Spot welds.

— `spotweld_technology`: Technology of the spot weld, see 9.2 Spot welds.

The element `<swop/>` does not allow any nested elements.

EXAMPLE

```
  <connection_0d label="RVT_2123921">
      <loc> 1645.83 821.145 616.585 </loc>
      <rivet head_diameter="8.5" head_height="0.9" hardness="410" shaft_diameter="5.4" sink_
size="0.3" length="1.5" >
            <normal_direction x="0" y="0" z="3" />
            <swop insert_shape="cone_23" insert_height="1.8" spotweld_diameter="4.5"
spotweld_technology="resistance" />
      </rivet>
      <appdata>
            ...
      </appdata>
  </connection_0d>
```

9.4.6 Clinch rivet studs

A clinch rivet stud is fixed to the base metal sheet, typically by cold forming, see Figure 17. This connection method does not need additional components. Special tools are used to plastically form a mechanical interlock between the pin and the sheet.

One or more panels, typically of different material, are attached to the stud and fastened using a counterpart (a coarse nut, or a Tucker plastic nut).

a) Clinched rivet stud (threaded) b) Clinched ball stud

Figure 17 — Clinch rivet studs — Threaded variant and ball stud

A clinch rivet stud is denoted by a nested element `<clinch_rivet_stud/>` within `<rivet/>`. This element is described completely by the attributes of both XML elements. The attributes `shaft_diameter`, `length`, and `part_code` are inherited from the `<rivet/>` element.

For the XML specification of `<clinch_rivet_stud/>` element, see Table 49:

Table 49 — Attributes of element `<clinch_rivet_stud/>`

Attributes	Type	Value space	Use	Constraint
press_in_force	Floating point	> 0,0	Optional	-

All attributes of this connection type are optional for importing it into CAD or CAE application.

These attributes have the following semantics:

— `press_in_force`: The force used to clinch the stud into the base sheet. For its unit, see 7.2.5 Unit system.

The element of `<clinch_rivet_stud/>` does not allow any nested elements.

The direction sense of `<normal_direction/>` is towards the base sheet, where the rivet penetrates the metal.

EXAMPLE

```
<connection_0d label="CNB_2123921">
    <loc> 1645.83 821.145 616.585 </loc>
    <rivet shaft_diameter="4.0" length="6.0" >
           <normal_direction x="0" y="0" z="3" />
           <clinch_rivet_stud press_in_force="2000"/>
    </rivet>
    <appdata>
           ...
    </appdata>
</connection_0d>
```

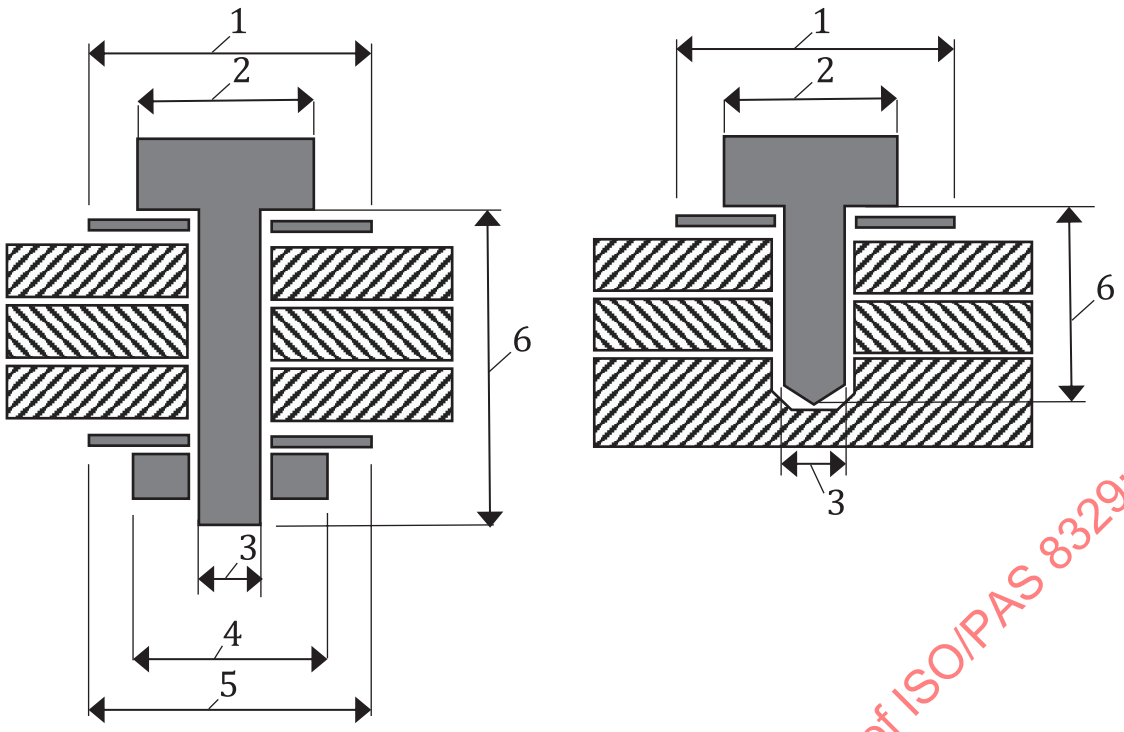
9.5 Threaded connections — Bolts and screws

9.5.1 General

Bolts and screws are probably the most well-known connection techniques. However, a closer look at their details is necessary. Screws and bolts are differentiated as follows (see also Figure 18):

— Bolts are used for assembling unthreaded components, with the aid of a nut.

— Screws are used in components which contain their own thread. The screw may even cut its own internal thread into them.



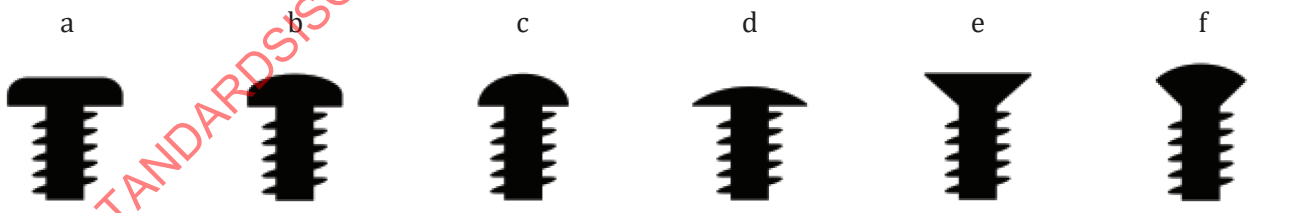
a) Bolt representation **b) Screw representation**

Key

1 head washer
2 head diameter
3 diameter
4 nut diameter
5 nut washer
6 length

Figure 18 — Bolts and screws

Figure 19 depicts different screw forms:

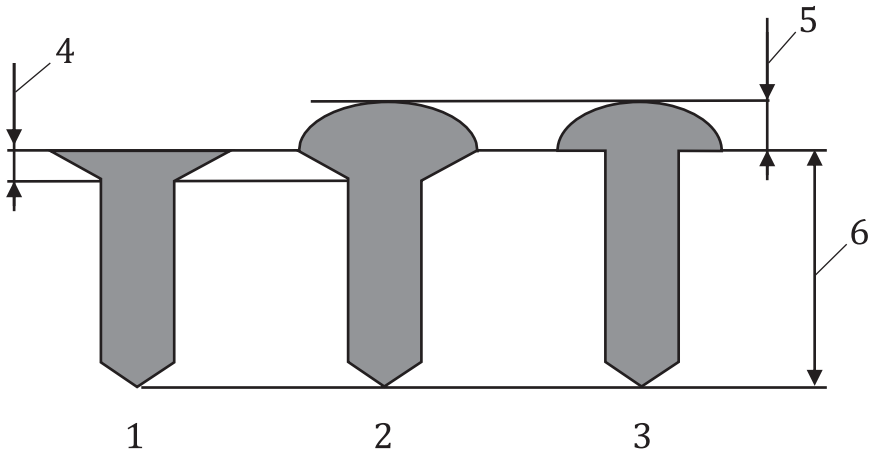


Button head **Pan head** **Round head** **Truss head** **Countersunk** **Countersunk (oval) Raised**

Key

Figure 19 — Different screw forms

Figure 20 explains the definition of length and head sizes:



Key

1 flat-head
2 oval-head
3 round-head
4 sink size
5 head height
6 length

Figure 20 — Definition of length and head sizes

Figure 21 visualizes the definition of lead, pitch and starts of a thread:

Key

1 lead
2 pitch

Figure 21 — Definition of lead, pitch and starts of a thread

9.5.2 Contacts and friction

Self-loosening of screws and bolts must be prevented. Static friction, together with pretension, is a major means against it. However, kinetic (or dynamic) friction also has some relevance in CAE processes. Therefore, both friction types need to be supported by χMCF.

The syntax for this has already been addressed in 7.4.3 .so the focus is on application details, now.

Friction occurs between any two objects in contact. In case of bolts and screws, these contacts are usually obtained between:

a) head and washer (a washer is loose if it is not fixed to the head or shaft),

b) washer (if there is one) and first connected part, or else,

c) head and first connected part [applicable only if not case b)],

d) the connected sheets,

e) last connected part and loose washer (if there is one),

f) washer (if there is one) and nut,

g) last connected part and nut [(applicable only if not case f)],

h) screw and cut thread, or bolt thread and nut thread.

Consequently, χMCF assigns friction attributes to:

— heads and nuts, applying to their contacts to either washers or adjacent parts,

— washers, applying to their contacts to adjacent parts (not counting head or nut),

— any contact between each two adjacent parts (addressed by 7.4.3.6),

— the thread (addressed by 9.5.3 below).

For heads (as a constituent part of a screw or bolt), nuts and washers, there are specific XML elements in χMCF. Corresponding friction attributes are located, there.

EXAMPLE 1 Bolted Joint with washer definition

```
<connection_group id="1">
   <connected_to>
      <part index="1" label="PART_7000400"/>
      <part index="2" label="PART_7100100"/>
      <part index="5" label="PART_5000300"/>
      <part index="6" label="PART_5000800"/>
   </connected_to>
   <connection_list>
     <connection_0d label="BOLT_135"> <!-- bolt with washers -->
       <loc> 84 60 10 </loc>
       <!-- Friction is "head to washer": -->
       <threaded_connection length="50"
                            static_friction="0.8"
                            thread_static_friction="0.8">
          <normal_direction x="0" y="0" z="-10"/>
          <!-- Washer next to head with its friction to 1st part -->
          <washer outer_diameter="20" attached="false" static_friction="0.8" />
          <bolt>
             <!-- Friction is "nut to washer" -->
             <nut diameter="16." static_friction="0.8">
               <!-- Washer next to nut with its friction to last part -->
               <washer outer_diameter="25" attached="false" static_friction="0.8" />
             </nut>
          </bolt>
       </threaded_connection>

       <contact_list>        <!-- Local Contact definition, according to clause 7.4.3.2 -->
           <contact>
               <partner part_index="1"/>
               <partner part_index="2"/>
               <coefficients static_friction="0.8"/>
           </contact>
           <contact>
               <partner part_index="2"/>
               <partner part_index="5"/>
               <coefficients static_friction="0.8"/>
           </contact>
           <contact>
               <partner part_index="5"/>
               <partner part_index="6"/>
```

```
                <coefficients static_friction="0.8"/>
            </contact>
        </contact_list>

      </connection_0d>
    </connection_list>
</connection_group>
```

EXAMPLE 2 Bolted Joint without washer definition but with global and local contact definition, plus thread contact. Local contacts override global contacts.

```
<connection_group id="1">

    <connected_to>
       <part index="1" label="PART_7000400"/>
       <part index="2" label="PART_7100100"/>
       <part index="5" label="PART_5000300"/>
       <part index="6" label="PART_5000800"/>
    </connected_to>

    <contact_list>   <!-- Global Contact Properties, for the whole connection_group -->
        <contact>
            <partner part_index="1"/>
            <partner part_index="2"/>
            <coefficients static_friction="0.8"/>
        </contact>
        <contact>
            <partner part_index="2"/>
            <partner part_index="5"/>
            <coefficients static_friction="0.8"/>
        </contact>
        <contact>
            <partner part_index="5"/>
            <partner part_index="6"/>
            <coefficients static_friction="0.8"/>
        </contact>
    </contact_list>

    <connection_list>
      <connection_0d label="BOLT_135"> <!-- bolt without washers -->
        <loc> 84 60 10 </loc>
        <!-- Friction "head to first part" and "thread to nut": -->
        <threaded_connection length="50"
                             static_friction="0.8"
                             thread_static_friction="0.8">
           <bolt>
              <!-- Friction is "nut to last part" -->
              <nut diameter="16." static_friction="0.8"/>
           </bolt>
        </threaded_connection>

        <contact_list>        <!-- Local Contact definition, according to clause 7.4.3.2 -->
            <contact>
              <partner part_index="1"/>
              <partner part_index="2"/>
              <coefficients static_friction="0.9"/>
            </contact>
        </contact_list>

      </connection_0d>
    </connection_list>

</connection_group>
```

9.5.3 Definition of element <threaded_connection/>

9.5.3.1 General

Due to their similar characters, bolts and screws share a couple of common attributes. To avoid redundancy, they are subsumed beneath a common, more abstract XML element <threaded_connection/>. Its nested elements are listed in Table 50:

Table 50 — Nested elements of <connection_0d/> for <threaded_connection/>

Nested elements	Multiplicity	Use	Constraints / Remarks
threaded_connection	1	Optional	-
Loc	1	Required	-
Appdata	1	Optional	See 7.3.2.
Femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

9.5.3.2 Element <loc/>

The syntax of this element is described in the corresponding 9.1.2 Location.

9.5.3.3 Element <appdata/>

This follows the syntax as defined in 7.3.2 User specific data <appdata/>.

9.5.3.4 Element <femdata/>

This follows the syntax as defined in 7.3.3 Finite element specific data <femdata/>.

9.5.3.5 Element <threaded connection/>

The XML specification of the <threaded_connection/> element is shown in Table 51:

Table 51 — Attributes of element <threaded_connection/>

Attributes	Type	Value space	Use	Constraints / Remarks
diameter	Floating point	> 0,0	Optional	
length	Floating point	> 0,0	Optional	-
thread_length	Floating point	> 0,0	Optional	length ≥ thread_length
head_diameter	Floating point	> 0,0	Optional	-
head_height	Floating point	≥ 0,0	Optional	If at least one of them is specified, *head_height + sink_size > 0* is required.
Head_type	Alphanumeric	Alphanumeric	Optional	-
sink_size	Floating point	≥ 0,0	Optional	Usually, sink_size > 0 implies no washer.
Pitch	Floating point	> 0,0	Optional	Not to be confused with “lead”.
Lead	Floating point	> 0,0	Optional	In case of single-start, thread form pitch is equal to lead. The default value is equal to pitch attribute.
Torque	Floating point	> 0,0	Optional	-
angle	Floating point	> 0,0	Optional	-
pretension	Floating point	> 0,0	Optional	-

Table 51 *(continued)*

Attributes	Type	Value space	Use	Constraints / Remarks
static_friction	Floating point	> 0,0	Optional	
kinetic_friction	Floating point	> 0,0	Optional	-
thread_static_friction	Floating point	> 0,0	Optional	
thread_kinetic_friction	Floating point	> 0,0	Optional	
strength_property_class	Alphanumeric	Alphanumeric	Optional	-
part_code	Alphanumeric	Alphanumeric	Optional	-

These attributes have the following semantics:

— `diameter`: the diameter of the bolt or screw. It should be provided, since, for example only few CAE simulation types can live without it.

— `length`: the length of the bolt or screw (see Figure 20).

— `thread_length`: the length of the thread of the bolt or screw. It is only needed in case of a partial-thread screw. In case of a full-thread screw, thread continues from tip to head, without a non-threaded area. In this situation, `thread_length = length - sink_size` applies.

— `head_diameter`: the diameter of the head of the bolt or screw.

— `head_height`: the height of the head.

— `head_type`: Type of screw head, e.g. “outer hexagonal”, “flanged-hex/Phillips-head combi”, “external torx plus”. Since there is a wide and ever-increasing variety of screw head types, an alphanumeric type is appropriate for this attribute.

— `sink_size`: the size of the head that is sunk (for countersunk screws).

— `pitch`: is the distance from the crest of one thread to the next (see Figure 21).

— `lead`: is the distance along the screw's axis that is covered by one complete rotation of the screw (360°). Lead and pitch are parametrically related by the number of starts (number of single thread). It very often is 1, in which case their relationship becomes equality. In general, lead is equal to S times pitch, where S is the number of starts.

— `torque`: The torque which should be applied when fastening the bolt or screw.

— `angle`: The turning angle which should be applied when fastening the bolt or screw.

— `pretension`: The pretension which is generated within the bolt or screw when fastening.

— `static_friction`: The static friction between head and adjacent washer or part.

— `kinetic_friction`: The kinetic friction between head and adjacent washer or part.

— `thread_static_friction`: The static friction between screw and cut thread, or bolt thread and nut thread.

— `thread_kinetic_friction`: The kinetic friction between screw and cut thread, or bolt thread and nut thread.

— `strength_property_class`: Strength according to applied standard within a unique part supplier or OEM.

— `part_code`: the part code of the bolt or screw, as used e.g. in a PDM system. It can be convenient to use the screw norm (according to e.g. ISO, EN, BSW, DIN) as part code.

Torque, pretension, and angle interact as follows:

— `torque` is only applied if no pretension is given.

— `angle` is only applied if torque is given, and no pretension is present.

For bolts as well as screws, it is recommended to provide the direction of fixation. Therefore, <threaded_connection/> offers following nested elements (see Table 52):

Table 52 — Nested elements of element <threaded_connection/>

Nested elements	Multiplicity	Use	Constraint
normal_direction	1	Optional	-
tangential_direction	1	Optional	-
bolt screw	1	Required	Exactly one of these elements.
Washer	1	Optional	-

9.5.3.6 Element <normal_direction/>

The direction of the bolt or screw is described by the element <normal_direction/> in the form of an orientation vector. This is necessary to define the orientation of the bolt or screw and therefore, which end is considered to be the connection's head-side. The orientation sense of the bolt is "from head to nut" and of the screw is "from head to point".

Refer to 9.1.3 for syntax of element <normal_direction/>.

Elements <bolt/>, <screw/> and <washer/> are described in the following subclauses.

The nested element <washer/> refers to the washer next to the head of a screw or bolt.

All attributes of threaded connections are optional for import to CAD or CAE processors. However, specific FE solvers may declare some of them to be mandatory.

General defaults are: 0 for numeric values, "" for strings. However, these defaults are not always useful for CAE.

9.5.4 Washer

Bolts and screws are frequently combined with washers. Therefore, we define the XML element <washer/>, see Table 53:

Table 53 — Attributes of element <washer/>

Attributes	Type	Value space	Use	Constraints / Remarks
outer_diameter	Floating point	> 0,0	Required	-
inner_diameter	Floating point	> 0,0	Optional	Usually NO inner diameter, if attached.
thickness	Floating point	> 0,0	Optional	-
attached	Boolean	"false" (default), "true"	Optional	-
static_friction	Floating point	> 0,0	Optional	-
kinetic_friction	Floating point	> 0,0	Optional	-
strength_property_class	Alphanumeric	Alphanumeric	Optional	-
part_code	Alphanumeric	Alphanumeric	Optional	NO part code, if attached.

These attributes have the following semantics:

— outer_diameter: the outer diameter of the washer. This value is mandatory.

— inner_diameter: the inner or hole diameter of the washer.

— thickness: the thickness of the washer.

— attached: true, if and only if the washer is firmly attached to the screw head or nut.

— `static_friction`: the static friction between this washer and its adjacent part (not head or nut).

— `kinetic_friction`: the kinetic friction between this washer and its adjacent part (not head or nut).

— `strength_property_class`: Strength according to applied standard within a unique part supplier or OEM.

— `part_code`: the part code of the washer, as used, e.g. in a PDM system. It can be convenient to use the washer norm as part code.

The element `<washer/>` does not allow for any nested elements.

9.5.5 Nut

Any bolt requires a nut. But since nuts may have several own attributes, χMCF defines a separate XML element for them (see Table 54)

Table 54 — Attributes of element `<nut/>`

Attributes	Type	Value space	Use	Constraints / Remarks
diameter	Floating point	> 0,0	Optional	-
height	Floating point	> 0,0	Optional	-
torque	Floating point	> 0,0	Optional	-
angle	Floating point	> 0,0	Optional	-
static_friction	Floating point	> 0,0	Optional	-
kinetic_friction	Floating point	> 0,0	Optional	-
clipped_to	Integer	> 0	Optional	-
fixed_to	Integer	> 0	Optional	-
strength_property_class	Alphanumeric	Alphanumeric	Optional	-
part_code	Alphanumeric	Alphanumeric	Optional	-

These attributes have the following semantics:

— `diameter`: the diameter of the nut.

— `height`: the height of the nut.

— `torque`: the torque which should be applied when fastening the nut.

— `angle`: the turning angle which should be applied when fastening the nut.

— `static_friction`: The static friction between the nut and the adjacent washer or part.

— `kinetic_friction`: The kinetic friction between the nut and the adjacent washer or part.

— `clipped_to`: If this attribute is given, the nut is fixed with a clip, or it is clinched, or it is a clip itself. It is clipped to the flange partner with the index given as attribute value (see 7.4.2.2). If this attribute is missing, the nut is not clipped. The nut and the clip share a common part code, i.e. they are regarded to be one single part;

— `fixed_to`: If this attribute is given, the nut is firmly fixed by welding or clinching to the flange partner with the index indicated by the attribute's value (see 7.4.2.2). If this attribute is missing, the nut is not fixed.

— `strength_property_class`: strength according to applied standard within a unique part supplier or OEM.

— `part_code`: the part code of the nut, as used, e.g. in a PDM system. It can be convenient to use the nut norm (according to e.g. ISO, EN, BSW, DIN) as part code.

Usually nut `fixed_to` prohibits nut `clipped_to` and vice versa.

Usually nut `clipped_to` or `fixed_to` prohibits bolt `clipped_to` or `fixed_to` and vice versa.

There are other means of fixating nuts to sheets as well, such as punching or riveting.

The element `<nut/>` allows for the following nested elements (see Table 55):

Table 55 — Nested elements of element `<nut/>`

Nested elements	Multiplicity	Use	Constraint
Washer	1	Optional	-

The nested element `<washer/>` refers to the washer next to the nut of the bolt.

9.5.6 Bolt

9.5.6.1 General

A bolt connection is denoted by an element `<bolt/>`. This element is described completely by its attributes and nested elements.

9.5.6.2 Element "bolt"

For the `<bolt/>` element, the following attributes can be specified (see Table 56):

Table 56 — Attributes of element `<bolt/>`

Attributes	Type	Value space	Use	Constraint
clipped_to	Integer	> 0	Optional	-
fixed_to	Integer	> 0	Optional	-

The following list explains these attributes:

— `clipped_to`: The head of the bolt is fixed with a clip to the flange partner with this index (see 7.4.2.2). If the attribute is missing, the bolt is not clipped. Both, bolt and clip, share a common part code, which means they are regarded to be one single part.

— `fixed_to`: The head of the bolt is fixed (e.g. welded) to the flange partner with this index (see 7.4.2.2). This also applies if there is no screw head at all, which means that this bolt actually is a fixed bolt, or a stud. If the attribute is missing, the bolt is not fixed.

There is no "base" attribute for bolts since this information can be derived from connection direction.

Usually bolt `fixed_to` prohibits bolt `clipped_to` and vice versa.

The element `<bolt/>` allows following nested elements (see Table 57):

Table 57 — Nested elements of element `<bolt/>`

Nested elements	Multiplicity	Use	Constraint
Nut	1	Optional	

The nested element `<nut/>` refers to the bolt's nut. This, in turn, may contain a nested element `<washer/>`.

The nested element `<nut/>` is required by the definition of a `<bolt/>`. The nut itself (respectively its `part_code` or property) is not allowed to be mentioned in element `<connected_to/>` of the `<connection_group/>` containing the `<bolt/>`. This allows keeping other connection types (e.g. glue, rivets) in the same `<connection_group/>`.

EXAMPLE 1

```
<connection_0d label="BOLT_100532">
    <threaded_connection diameter="10.0" length="50.0"
        head_diameter="16.0" head_height="5" sink_size="3">
        <normal_direction> x="3.0" y="0.0"  z="0.0"/>
        <!-- magnitude is irrelevant, direction sense is from head to nut -->
        <bolt>
        <nut diameter="16." height="5">
            <washer outer_diameter="20"/>
        </nut>
        </bolt>
        <washer outer_diameter="20">
    </threaded_connection>
    <loc> 1500.3 838.7 730.6 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

EXAMPLE 2

```
<connection_0d label="BOLT_135">
    <threaded_connection diameter="10.0" length="50.0"
         head_diameter="16.0" head_height="5" thread_length="35"
         torque="80" angle="30" pretension="1200" part_code="M10x50 12.9" >
        <normal_direction x="0" y="0" z="-10"/>
        <!-- Washer next to head -->
        <washer outer_diameter="20" inner_diameter="10.3" thickness="1.5"
             attached="false" part_code="M10x20x1.5"/>
        <bolt fixed_to="1" >
            <nut diameter="16." height="5" static_friction="0.8">
                <!-- Washer firmly attached to nut -->
                <washer outer_diameter="25" thickness="1.5" attached="true"/>
            </nut>
        </bolt>
    </threaded_connection>
        <loc> 1500.3 838.7 730.6 </loc>
    <appdata>
            ...
    </appdata>
</connection_0d>
```

EXAMPLE 3

```
<connection_0d label="BOLT_135">
    <threaded_connection length="50" diameter="10"
         head_diameter="16" head_height="5" thread_length="35"
         torque="80" angle="30" pretension="1200" part_code="M10x50 12.9">
        <normal_direction x="0" y="0" z="-10"/>
        <!-- Washer is part of the head, so it cannot have part code -->
        <washer outer_diameter="20" inner_diameter="10.3" thickness="1.5"
            attached="true"/>
        <bolt>
            <nut diameter="16." height="5" static_friction="0.8" clipped_to="4"/>
        </bolt>
    </threaded_connection>
    <loc> 1500.3 838.7 730.6 </loc>
    <appdata>
            ...
    </appdata>
</connection_0d>
```

EXAMPLE 4 Bolted joint with maximum parameter usage

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<xmcf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="xmcf_3_1_1.xsd">
   <version> 3.1.1 </version>
```

```
    <date> 2016-01-08 </date>
    <units length="mm" angle="deg" mass="kg" force="N" torque="Nm" time="s"/>
    <connection_group id="1">
       <connected_to>
          <part index="1" label="PART_7000400"/>
          <part index="2" label="PART_7100100"/>
          <part index="5" label="PART_5000300"/>
          <part index="6" label="PART_5000800"/>
       </connected_to>
       <connection_list>
          <connection_0d label="BOLT_135">   <!-- bolt with washers -->
             <loc> 84 60 10 </loc>
             <!-- Friction between "head to washer" and " thread and nut ": -->
              <threaded_connection diameter="10" length="50" thread_length="26"
                head_diameter="16" head_height="6.4" head_type="hexagonal"
                sink_size="0" pitch="0.75" lead="1.5"
                torque="20" angle="35" pretension="180"
                static_friction="0.8" kinetic_friction="0.6"
               thread_static_friction="0.6"
                strength_property_class="8.8"
                part_code="M10x50 8.8">
                <normal_direction x="0" y="0" z="-10"/>
                <!-- Washer next to head with its friction to 1st part -->
                <washer outer_diameter="20" inner_diameter="10.4" thickness="1.25"
                   attached="false"
                   static_friction="0.8" kinetic_friction="0.6"
                   strength_property_class="8.8"
                   part_code="W20/10.4x1.25 8.8"/>
                <bolt>
                   <!-- No Friction nut to washer, since washer is attached! -->
                   <nut diameter="16." height="6.4"
                      torque="20" angle="35"
                      clipped_to="6"
                      strength_property_class="8.8"
                      part_code="N10 8.8">
                      <!-- Washer attached to nut with its friction to last part -->
                      <washer outer_diameter="25" attached="true" static_friction=".8"/>
                   </nut>
                </bolt>
             </threaded_connection>

             <contact_list>      <!-- friction between adjacent flange partners -->
                 <contact>
                     <partner part_index="1"/>
                     <partner part_index="2"/>
                     <coefficients static_friction="0.8"/>
                 </contact>
                 <contact>
                     <partner part_index="2"/>
                     <partner part_index="5"/>
                     <coefficients static_friction="0.8"/>
                 </contact>
                 <contact>
                     <partner part_index="5"/>
                     <partner part_index="6"/>
                     <coefficients static_friction="0.8"/>
                 </contact>
             </contact_list>

          </connection_0d>
       </connection_list>
    </connection_group>
</xmcf>
```


9.5.6.3 Possible bolt and screw assemblies

χMCF recognizes the following mounting constellations:

a) Bolt with welded nut (to the bottom sheet), see Figure 22:

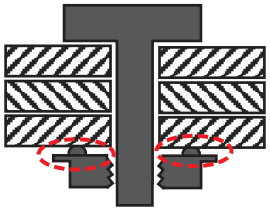


Figure 22 — Bolt with welded nut

EXAMPLE 1

```
<connection_0d label="BOLT_135">
    <threaded_connection diameter="10.0" length="50.0"
         head_diameter="16.0" head_height="5" thread_length="35"
         torque="80" angle="30" pretension="1200" part_code="M10x50 12.9" >
        <normal_direction x="0" y="0" z="-10"/>
        <!--No Washer in this case-->
        <bolt>
            <nut diameter="16." height="5" fixed_to="3" />
        </bolt>
    </threaded_connection>
    <loc> 1500.3809  838.75885  730.6529 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

b) Bolt with clipped nut (clipped to the bottom sheet): This is the same constellation, only `fixed_to` is replaced by `clipped_to`.

c) Bolt with free nut (not clipped, nor welded to the bottom sheet), see Figure 23:

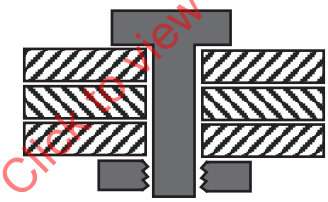


Figure 23 — Bolt with free nut

NOTE Since both the screw and the nut are free, there is no `fixed_to` nor `clipped_to` attribute.

d) Screw (screwed to the last sheet), see Figure 24:

Figure 24 — Screw

EXAMPLE 2

```
<connection_0d label="SCREW_139">
    <threaded_connection diameter="10.0" length="50.0"
         head_diameter="16.0" head_height="5" thread_length="35"
         torque="80" angle="30" pretension="1200" part_code="M10x50 12.9" >
        <normal_direction x="0" y="0" z="-10"/>
        <screw base="3"/>
```

```
    </threaded_connection>
    <loc> 1500.3809  838.75885  730.6529 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

e) Welded stud (with a free nut, of course), see Figure 25:

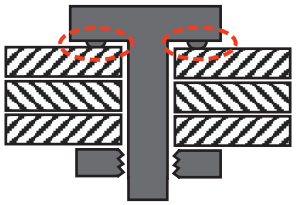


Figure 25 — Welded stud with free nut

EXAMPLE 3

```
<connection_0d label="BOLT_135">
    <threaded_connection diameter="10" length="50" head_diameter="16" head_height="5"
       thread_length="35" torque="80" angle="30" pretension="1200"
       part_code="M10x50 12.9">
        <normal_direction x="0" y="0" z="-10"/>
        <!--No Washer in this case-->
        <bolt fixed_to="1" >
            <nut diameter="16." height="5" />
        </bolt>
    </threaded_connection>
    <loc> 1500.3809  838.75885  730.6529 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

f) Plain stud (with a nut on one end, screwed into a part on the opposite end), see Figure 26:

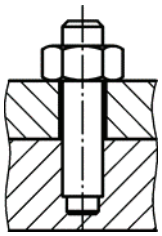


Figure 26 — Plain stud

These studs are not a feature of χMCF version 3.1.1 and below. They can be modelled according to case d) but may become a topic of a future χMCF version.

In all cases, the `<connected_to/>` element contains only the assemblies, part codes or PIDs of the connected sheets.

9.5.7 Screw

9.5.7.1 General

A screw connection is denoted by an element `<screw/>`. This element is described completely by its attributes and nested elements.

9.5.7.2 Element “screw”

For the `<screw/>` element, the following attributes can be specified (see Table 58):

Table 58 — Attributes of element `<screw/>`

Attributes	Type	Value space	Use	Constraint
base	Integer	> 0	Optional	-

— `base`: the index (see 7.4.2.2) of the flange partner, which is carrying the thread. If the attribute is missing, the threaded part has to be derived from the connection direction.

Specific subtypes of screws are defined by adding related nested elements, listed in the following Table 59:

Table 59 — Nested elements of element `<screw/>`

Nested elements	Multiplicity	Use	Constraint
flow_drilled	1 - *	Optional	-

The subtypes are described in detail in the subclauses below.

EXAMPLE 1 Screw without attributes

```
<connection_0d label="SCREW_100532">
      <threaded_connection length="50." diameter="10"
           head_diameter="16." head_height="3" sink_size="4">
           <normal_direction x="3.0" y="0.0" z="0.0" />
           <!-- magnitude is irrelevant, direction sense is from head to point -->
           <screw /> <!-- Screw may come without any attributes -->
           <washer outer_diameter="20"/>
      </threaded_connection>
      <loc> 1500.3809 838.75885 730.6529 </loc>
      <appdata>
          ...
      </appdata>
</connection_0d>
```

EXAMPLE 2 Screw with “base” attribute and with washer)

```
<connection_0d label="SCREW_100532">
    <threaded_connection length="50" diameter="10"
          head_diameter="16" head_height="5" thread_length="35">
          <normal_direction x="0" y="0" z="-10"/>
          <washer outer_diameter="20" inner_diameter="10.3"/> <!--Washer next to head-->
          <screw base="5" />
    </threaded_connection>
    <loc> 1500.3809 838.75885 730.6529 </loc>
    <appdata>
          ...
    </appdata>
</connection_0d>
```

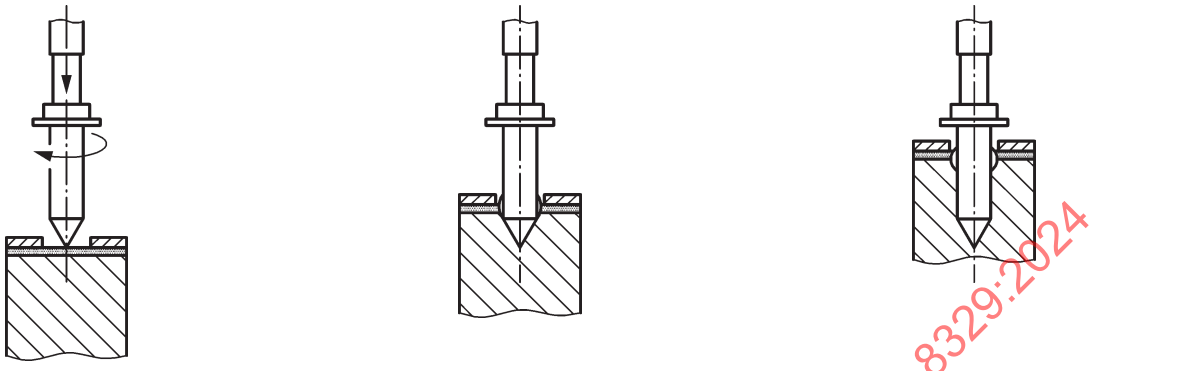
EXAMPLE 3 Screw with attributes but without washer

```
<connection_0d label="SCREW_100532">
    <threaded_connection length="50" diameter="10"
          head_diameter="16" head_height="5" sink_size="1" thread_length="35" >
          <normal_direction x="0" y="0" z="-10"/>
          <screw base="5" />
    </threaded_connection>
    <loc> 1500.3809 838.75885 730.6529 </loc>
    <appdata>
          ...
    </appdata>
</connection_0d>
```

9.5.7.3 Flow drilled screws (FDSs)

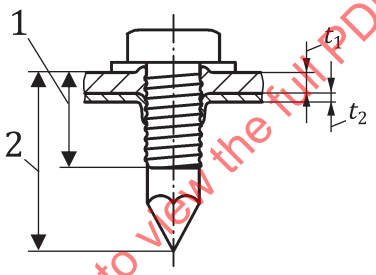
9.5.7.3.1 General

A flow drilled screw (FDS) is applied by a process called “friction drilling”, see Figure 27 and Figure 28.



a) Placing screw (with speed and pressure) **b) Heating, forming material** **c) Forming thread and tighten screw**

Figure 27 — Process of flow drilled screwing (FDS)



Key

1 thread length

2 length

Figure 28 — Measures of an applied flow drilled screw (FDS)

The basic steps in the FDS process are:

a) applying rotational velocity and pressure,

b) heating the target sheet metal (or without pre-punching both sheet component) by the tool and melts it through,

c) tapping the screw thread,

d) tightening the screw and applying proper torque to create the desired connection.

The FDS combines the tool with the screw. The screw itself drills its hole and shapes its thread.

9.5.7.3.2 Element “flow_drilled”

For the `<flow_drilled/>` element, the following attributes can be specified (see Table 60):

Table 60 — Attributes of element `<flow_drilled/>`

Attributes	Type	Value space	Use	Constraint
pre_machined_hole_diameter	Floating point	≥ 0,0	Optional	-
pre_machined_hole_index	Integer	> 0	Optional	Exists only if `<connected_to/>` properly filled out with parts to be connected.
pilot_hole_diameter	Floating point	≥ 0,0	Optional	Its definition depends on the applied FDS type.

— `pre_machined_hole_diameter`: In order to facilitate the penetration of the metal sheet by the tip of the FDS, a small hole may be machined in the sheet metal. Furthermore, when the penetration happens in the phase of material forming, a small portion of the formed part flows opposite to the fastening direction and creates a bulge (d_W) that has to be accommodated by the clearance-hole (d_D). The default value is 0,0, which means “no pre-machined hole or clearance hole”, see Figure 29:

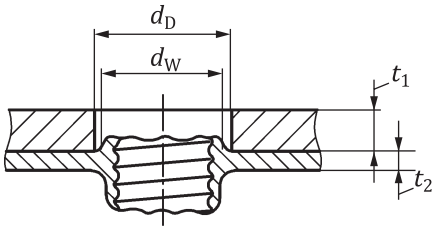


Figure 29 — FDS connection with pre-machined clearance hole

— `pre_machined_hole_index`: If `pre_machined_hole_diameter` > 0,0, then the hole is in the flange partner with index `pre_machined_hole_index` (see 7.4.2.2). If the attribute is missing, this information is not (yet) available.

— `pilot_hole_diameter`: This hole diameter (d_V) is defined in case the applied FDS type requires a drilled hole on the sheet metal that is to be formed during the process, see Figure 30:

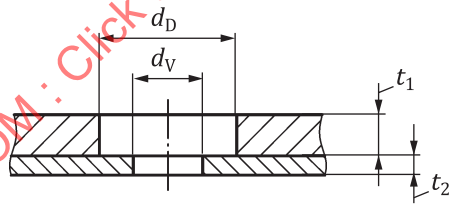


Figure 30 — Pilot hole on sheet metal

The element `<flow_drilled/>` does not allow for any nested elements.

EXAMPLE

```
<connection_0d label="FDS_96930">
    <threaded_connection length="50" diameter="10"
      head_diameter="16" head_height="5" sink_size="1" thread_length="35" >
        <normal_direction x="0" y="0" z="-10"/>
        <screw base="1">
            <flow_drilled pre_machined_hole_diameter="18.0"
              pre_machined_hole_index="1" pilot_hole_diameter="12.0" />
        </screw>
    </threaded_connection>
    <loc> 1500.3809 838.75885 730.6529 </loc>
    <appdata>
         ...
    </appdata>
</connection_0d>
```

9.6 Gum drops

A gum drop, or adhesive point, is denoted by an element `<gumdrop/>`. This element is described completely by its attributes and nested elements, see Table 61.

Table 61 — Nested elements of `<connection_0d>` for `<gumdrop/>`

Nested elements	Multiplicity	Use	Constraints / Remarks
gumdrop	1	Optional	-
Loc	1	Required	-
appdata	1	Optional	See 7.3.2.
femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

The XML specification of `<gumdrop/>` is shown in the following Table 62:

Table 62 — Attributes of element `<gumdrop/>`

Attributes	Type	Value space	Use	Constraint
Diameter	Floating point	>= 0,0	Optional	-
Mass	Floating point	>= 0,0	Optional	-
Material	Alphanumeric	Alphanumeric	Optional	-

The following list explains the attributes:

— `diameter`: The diameter of a gumdrop is specified by the attribute `diameter` for the child element of `<connection_0d/>`. It specifies the diameter of the adhesive material after manufacturing.

— `mass`: the mass of the glue attached.

— `material`: the name of the adhesive material according to CAD/PDM. For CAE applications, another label from a reduced data base may be applicable. This can be stored in this case in `<appdata/>` or `<custom_attributes/>`, see 8.6.

The element `<gumdrop/>` allows for following nested elements (see Table 63):

Table 63 — Nested elements of element `<gumdrop/>`

Nested elements	Multiplicity	Use	Constraint
normal_direction	1	Optional	-
tangential_direction	1	Optional	-

EXAMPLE

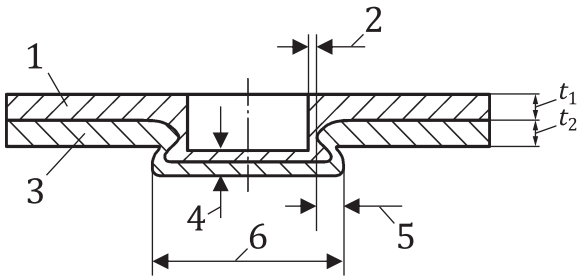
```
<connection_0d label="DROP_2123921">
    <!-- Assumed Unit system with mass attribute with value="kg" -->
    <gumdrop diameter="5.0" mass="0.0033" material="CAD_Material" />
    <loc> 1645.83 821.145 616.585 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

9.7 Clinches

Clinching is a mechanical, cold forming fastening method to join sheet metal without additional components, using special tools to plastically form a mechanical interlock between the sheets.

In general, clinching is used for light metal materials, as these can only be welded in poor quality or not at all. This joining technique can also be a cost-effective alternative to spot welding for specific steel structures. Such joints can typically be found on air conditioning tube fixations or air bag assemblies.

As a result, the cross-section of a clinch can look as shown in Figure 31:

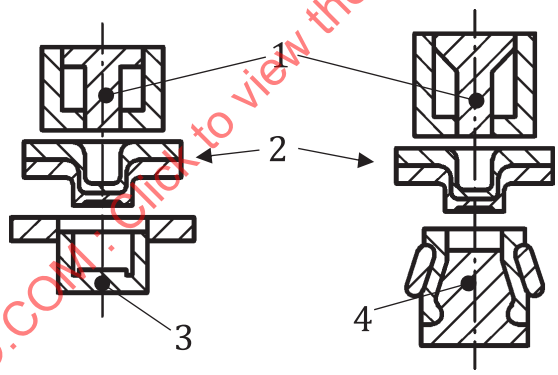


Key

1 punch side material
2 neck
3 die side material
4 cap thickness
5 interlock
6 button diameter

Figure 31 — Clinch joint dimensions

Figure 32 illustrates typical tool arrangements around a to-be clinch:



Key

1 punch
2 clinch joint
3 fixed die
4 openable die

Figure 32 — Two example clinch systems[12] (TOX (left) and BTM's Tog-L-Loc system)

If such a cross-section is rotated around its vertical axis, a pan-shaped round clinch results in three dimensions. Alternatively, this cross-section can be regarded as the view at an open edge of two stacked sheets. The shape's height reduces, as the section proceeds "behind the paper", resulting in a wedge-shaped 3-dimensional contour.

A wide range of geometrical shapes, produced by as many different tools, is possible. Therefore, an enumeration of all clinches cannot be provided. They shall be described by OEM-specific alphanumeric names. The same is valid for the strength of the clinch, in terms of its strength class.

A clinch is denoted by an element <clinch/>. This element is described completely by its attributes and nested elements, see Table 64 and Table 65:

Table 64 — Nested elements of <connection_0d/> for <clinch/>

Nested elements	Multiplicity	Use	Constraints / Remarks
clinch	1	Optional	-
loc	1	Required	-
appdata	1	Optional	See 7.3.2.
femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

The XML specification of the <clinch/> element is shown in Table 65:

Table 65 — Attributes of element <clinch/>

Attributes	Type	Value space	Use	Constraints / Remarks
clinch_type	Alphanumeric	Alphanumeric	Optional	-
strength_class	Alphanumeric	Alphanumeric	Optional	It is dependent from the applied punch diameter and part materials
shear_strength	Floating point	> 0,0	Optional	-
peel_strength	Floating point	> 0,0	Optional	-
button_diameter	Floating point	> 0,0	Optional	Dependent of punch diameter and sheet thicknesses
die_type	Alphanumeric	Alphanumeric	Optional	"round" or "rectangular"

The following list explains the attributes:

— clinch_type: the alphanumeric name of the clinch. This document refers to two systems which are called "TOX" and BTM's Tog-L-Loc or Lance-N-Loc[12] system. The main difference is that the TOX system uses a fixed die, whereas the BTM system employs an extending die (see Figure 32). For more process and system details, refer to the documentation and website information of the specific clinch equipment supplier.

— strength_class: the strength class name of the clinch. Since the manufacturer of the applied clinching process has a specific tooling die diameter, three different strength classes can be defined, such as:

  — Heavy Duty (HD) punches are 6,4 mm/0,25" in diameter and are used for material up to 0,35 mm/0,135" thick. A HD joint is typically twice as strong as an equivalent MD joint.

  — Medium Duty (MD) punches are the most common and are approx. 4,6 mm/0,18" in diameter and are used for materials which are between 0,20 mm/0,075" and 0,025 mm/0,010" thick.

  — Light Duty (LD) punches are 3,0 mm/0,12" in diameter and are used for materials up to 0,08 mm/0,032" thick. LD joints are typically half as strong as a MD joint.

— shear_strength: Shear failure where the joint fails by shearing a hole in the punch side material. It is defined as maximum measured force during the test process.

— peel_strength: Pull failure in peeling test is where the joint pulls apart leaving "male" and "female" parts. It is defined as maximum measured force during the test process.

— button_diameter: The applied button diameter to create this joint. The following example formula can be used: $D_{button} = d_{nom} \times 1{,}4$, where d_{nom} is the punch diameter.

— die_type: The "round" dies (three and four blades) are used for drawable materials (like mild steel and aluminium). The "rectangular" dies (two blades) are used for hard materials (materials that do not draw very well) such as stainless steel.

If possible, a clinch should know the direction of fixation, therefore, possess a nested element `<normal_direction/>`. However, this is not mandatory in order to allow for importing incomplete data. Direction sense of `<normal_direction/>` is from punch to die, which represents the direction in which metal is displaced. The direction element definition can be found in 9.1.3.

There is no “base” attribute for clinches since this information can be derived from connection direction.

The element `<clinch/>` allows for following nested elements, see Table 66:

Table 66 — Nested elements of element `<clinch/>`

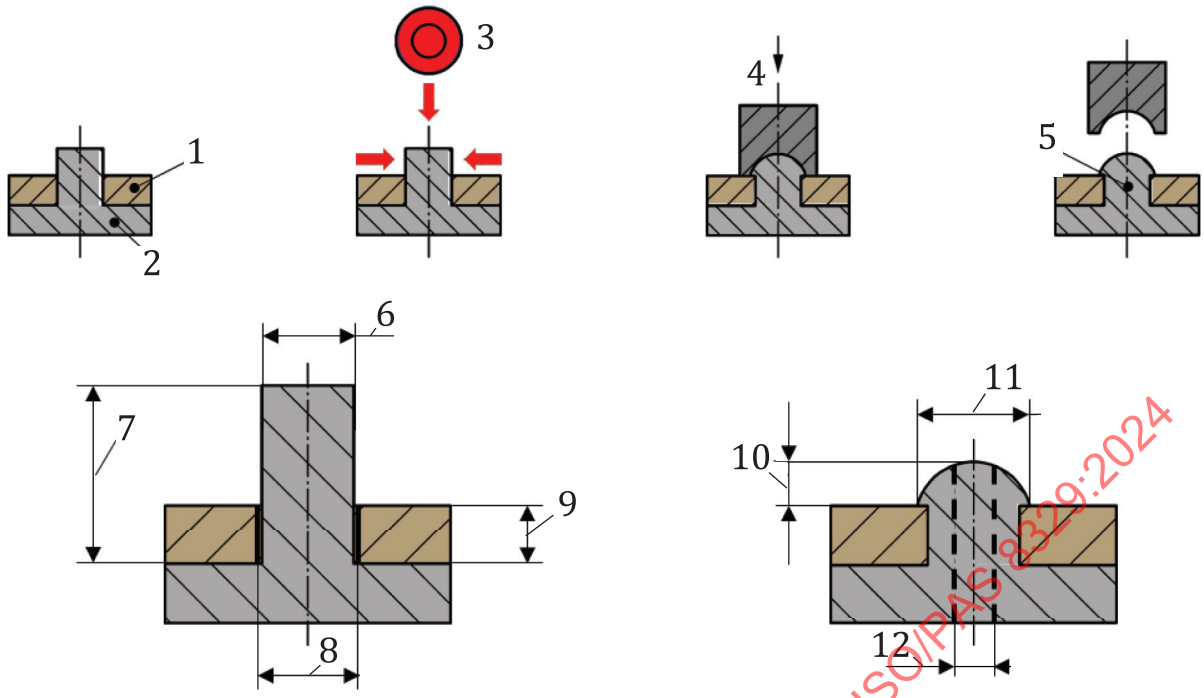
Nested elements	Multiplicity	Use	Constraint
normal_direction	1	Optional	-
tangential_direction	1	Optional	-

EXAMPLE

```
<connection_0d label="CLINCH_left_2123521">
    <!-- Unit definition and connected to is important for clinch -->
    <clinch clinch_type="TOX" button_diameter="3.0"
      strength_class="HD" shear_strength="890" peel_strength="356">
        <normal_direction x="0" y="0" z="-10"/>
    </clinch>
    <loc> 1645.83 821.145 616.585 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

9.8 Heat stakes / Thermal stakes

Heat stakes are a well-known joint type to connect a shell-type part with a thermoplastic other part. For this purpose, the thermoplastic part is manufactured with appropriate stakes, see Figure 33:



Key

1 joined material
2 thermoplastic
3 heat
4 forming
5 form-closed
6 diameter
7 boss height
8 hole diameter (>D)
9 captured material thickness (T)
10 head_height
11 head_diameter (=2D)
12 void diameter (optional)

Figure 33 — Heat stakes — Process steps and design dimensions

Rotation of this cross-section around its vertical axis yields a round shape in three dimensions. This shape is the most common, although not mandatory. A wide range of other geometrical shapes, produced by as many different tools, is possible.

Therefore, an enumeration of all heat stake types cannot be provided. They shall be described by OEM specific alphanumeric names (e.g. flared, domed, knurled, hollow, flush). The same is valid for the strength of the connection, in terms of its force-displacement diagram.

Heat stakes cannot be disassembled without irreversible damage to (at least) the thermoplastic part.

The element `<heat_stake/>` is described completely by its attributes and nested elements (see Table 67):

Table 67 — Nested elements of `<connection_0d/>` for `<heat_stake/>`

Nested elements	Multiplicity	Use	Constraints / Remarks
heat_stake	1	Optional	-
loc	1	Required	-
appdata	1	Optional	See 7.3.2.
femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

The XML specification of the `<heat_stake/>` element is shown in Table 68:

Table 68 — Attributes of element `<heat_stake/>`

Attributes	Type	Value space	Use	Constraints / Remarks
heat_stake_type	Alphanumeric	Alphanumeric	Optional	-
strength	Floating point	> 0,0	Optional	-
diameter	Floating point	> 0,0	Optional	diameter < hole_diameter
head_diameter	Floating point	> 0,0	Optional	-
head_height	Floating point	≥ 0,0	Optional	-
void_diameter	Floating point	≥ 0,0	Optional	void_diameter < diameter
hole_diameter	Floating point	> 0,0	Optional	hole_diameter < head_diameter

The following list explains the attributes:

— `heat_stake_type`: the alphanumeric name of the heat stake (e.g. domed, flared),

— `strength`: the strength of the heat stake,

— `diameter`: the diameter of the heat stake, assuming a round/cylindrical shape,

— `head_diameter`: the diameter of the head of the heat stake after thermal forming, assuming the final shape is round,

— `head_height`: the height of the head, as created by the tool,

— `void_diameter`: The tool may form a hole/void within the stake. This is its diameter, assuming cylindrical shape.

— `hole_diameter`: Diameter of the hole(s) in the non-thermoplastic part(s).

If possible, a heat stake should know the direction of fixation, therefore, possess a nested element `<normal_direction/>`. However, this is not mandatory in order to allow for importing of incomplete data. Direction sense of `<normal_direction/>` is from tool to thermoplastic part. The direction element definition can be found in 9.1.3.

There is no "base" attribute for heat stakes since this information can be derived from the connection direction.

The initial height of the stake (above base part) is not represented in χMCF: Before tool application, it can be derived from CAD data. After tool application (in final shape of the heat stake), this height has vanished.

The element `<heat_stake/>` allows for following nested elements, see Table 69:

Table 69 — Nested elements of element `<heat_stake/>`

Nested elements	Multiplicity	Use	Constraint
normal_direction	1	Optional	-
tangential_direction	1	Optional	-

EXAMPLE

```
<connection_0d label="HEAT_STAKE_521">
    <heat_stake heat_stake_type="domed" diameter="3.0"
      head_diameter="6.0" head_height="2.25">
        <normal_direction x="0" y="0" z="-10"/>
    </heat_stake>
    <loc> 1645.83 821.145 616.585 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

9.9 Clips / Snap joints

In general, a clip is a fastener with an elastic component. Pushed onto a firm counterpart, this elastic component causes the clip to hook onto that part. Depending on the type of the clip, it can be removed without being destroyed.

A wide and ever-increasing variety of clinches is in practical use. Examples are:

— A “Terry Clip” consists of a cylindrical metal band with a gap. Opening the gap, it snaps onto a tube. Frequently, there are means for fastening a screw etc. on the opposite side of the gap.

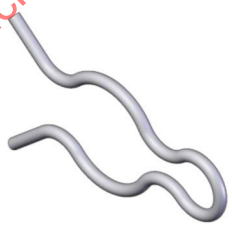
— A “Hairpin Clip” (see Figure 34) is similar to a “Terry Clip” but uses some wire instead of a metal band.

— An “R-Clip” resembles a “Hairpin Clip”, but one of its legs is straight and suitable for inserting into a drilled hole of an axle.

— A “Circlip” (see Figure 35, also known as a C-Clip, Seeger ring, snap ring, or Jesus clip) is used to secure some item against sliding on an axle.

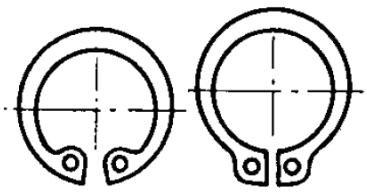
— Another sort of clips is snapped into a hole in a sheet metal (see Figure 36). Its other side is shaped to hold a certain item, e.g. a cable or a panel.

— Other clips slide onto a flat surface (see Figure 37).



NOTE Source: Wikimedia Commons, 2009-08-19 by Wizard191, CC BY-SA 3.0, https://creativecommons.org/licenses/by-sa/3.0/

Figure 34 — Hairpin clip



NOTE Source: Wikimedia Commons, 2004-09-06, both by Jean-Jacques MILAN at French Wikipedia, public domain, via Wikimedia Commons.

Figure 35 — Internal (left) and external (right) circlips

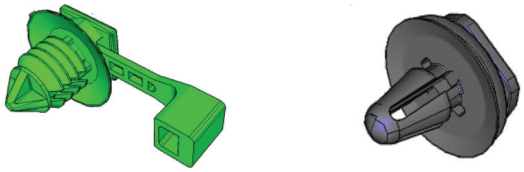


Figure 36 — Clips pushed into a hole

Figure 37 — Clips sliding onto a flat surface

A clip is denoted by an element `<clip/>` and described completely by its attributes and nested elements, see Table 70:

Table 70 — Nested elements of `<connection_0d/>` for `<clip/>`

Nested elements	Multiplicity	Use	Constraints / Remarks
Clip	1	Optional	-
loc	1	Required	-
appdata	1	Optional	See 7.3.2.
femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

The XML specification of the `<clip/>` element is shown in Table 71:

Table 71 — Attributes of element `<clip/>`

Attributes	Type	Value space	Use	Constraints / Remarks
clip_type	Alphanumeric	Alphanumeric	Optional	-
attachment_type	Alphanumeric	Alphanumeric	Optional	-
hole_diameter	Floating point	≥ 0,0	Optional	-
hole_length	Floating point	≥ 0,0	Optional	hole_length > 0 implies hole_diameter > 0
pin_diameter	Floating point	≥ 0,0	Optional	-
pin_width	Floating point	≥ 0,0	Optional	pin_width > 0 implies pin_diameter > 0
pin_length	Floating point	≥ 0,0	Optional	-
strap_length	Floating point	≥ 0,0	Optional	-
clipped_to	Integer	> 0	Optional	-
material	Alphanumeric	Alphanumeric	Optional	-
part_code	Alphanumeric	Alphanumeric	Optional	-

The following list explains the attributes:

— `clip_type`: the alphanumeric name of the clip, e.g. “STRAP 5-45X8X.9-4.1 PNL”,

— `attachment_type`: the description, how the clip is fastened, e.g. “push into round hole”,

— `hole_diameter`: If the clip is pushed into a hole, this attribute describes the diameter of that mating hole. If the hole is not round, the minimum diameter is meant. The default value is 0,0, which means “no hole”,

— `hole_length`: If the clip is pushed into a non-round hole, this attribute describes the maximum diameter of that hole. The default value is 0,0, which means “no hole or round hole”,

— `pin_diameter`: If the clip is pushed into a hole, this attribute describes the diameter of the clip’s pin. If the hole is not round, the minimum diameter is meant. The default value is 0,0, which means “no hole”,

— `pin_width`: If the clip is pushed into a non-round hole, this attribute describes the maximum diameter of the clip’s pin. The default value is 0,0, which means “no hole or round hole”,

— `pin_length`: If the clip is pushed into a hole, this attribute describes the length of the clip’s pin. The default value is 0,0, which means “no hole”,

— `strap_length`: If the clip carries a strap (see Figure 36 — Clips pushed into a hole, left picture), this attribute describes the length of that strap. The default value is 0,0, which means “no strap”,

— `clipped_to`: The clip is clipped to the flange partner with this index (see 7.4.2.2). If attribute is missing, this information is not (yet) available,

— `material`: the material of the clip,

— `part_code`: the part code of the clip, as used e.g. in a PDM system.

There is no `base` attribute for clips since this information is hold by attribute `clipped_to`.

If possible, a clip should know the direction of fixation, i.e. have a nested element `<normal_direction/>`. However, this is not mandatory in order to allow for importing incomplete data. The direction sense of `<normal_direction/>` is from tool to the flange partner given by attribute `clipped_to`.

Element `<tangential_direction/>` denotes direction of (one) maximum clip diameter, perpendicular to `<normal_direction/>`. This gives the local x axis. The `<normal_direction/>` and `<tangential_direction/>` elements are described in 9.1.3.

The element `<clip/>` allows for following nested elements (see Table 72):

Table 72 — Nested elements of element `<clip/>`

Nested elements	Multiplicity	Use	Constraint
normal_direction	1	Optional	-
tangential_direction	1	Optional	-

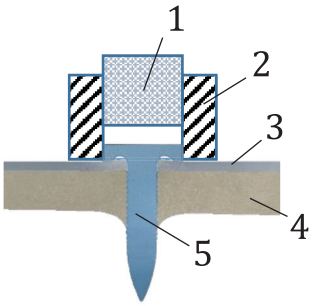
EXAMPLE

```
  <connection_0d label="CLIP_1001">
      <clip clipped_to="1" attachment_type="push into round hole" hole_diameter="8.0" hole_
length="12.0" pin_diameter="10.0" pin_length="10.0" material="polyamid">
          <normal_direction x="0" y="0" z="-10"/>
          <tangential_direction x="0" y="10" z="0"/>
      </clip>
      <loc> 1645.83 821.145 616.585 </loc>
      <appdata>
          ...
      </appdata>
  </connection_0d>
```

9.10 Nails

Nailing is a rather old joining method. However, with optimized nail shapes and high velocity application, it still addresses modern requirements, especially if non-steel materials are involved. The components, which

are connected by this type of connector, may consist of steel, aluminium, magnesium, or plastic. Figure 38 shows an example of the cross-section of an applied nail:



Key

1 punch
2 blank holder
3 joined sheet
4 base part
5 nail

Figure 38 — Cross-section of a nail joint connecting two sheets

A nail is denoted by an element `<nail/>`. This element is described completely by its attributes and nested elements, see Table 73:

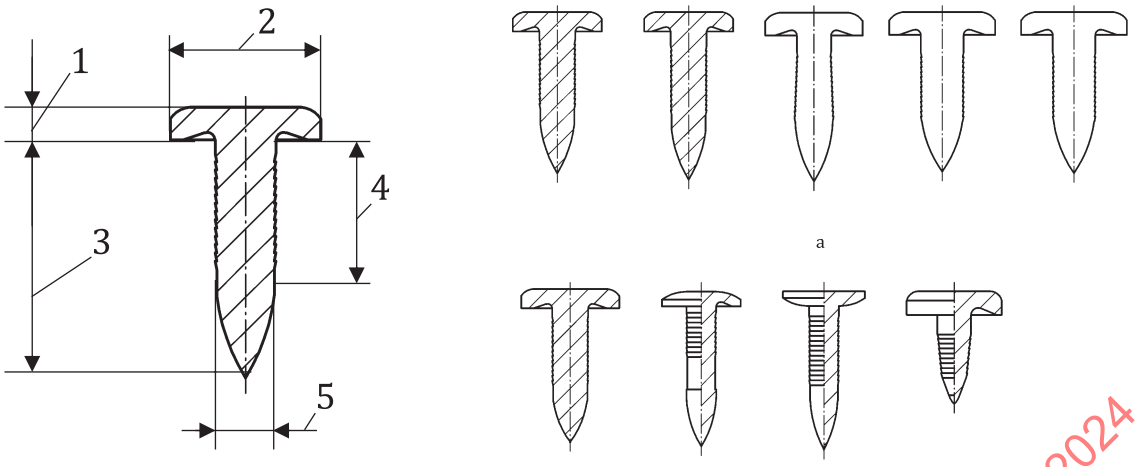
Table 73 — Nested elements of `<connection_0d/>` for `<nail/>`

Nested elements	Multiplicity	Use	Constraint / Remarks
nail	1	Optional	-
loc	1	Required	-
appdata	1	Optional	See 7.3.2.
femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

The XML specification of the `<nail/>` element is shown in Table 74:

Table 74 — Attributes of element `<nail/>`

Attributes	Type	Value space	Use	Constraint / Remarks
nail_type	Alphanumeric	Alphanumeric	Optional	-
shaft_diameter	Floating point	> 0,0	Optional	-
length	Floating point	> 0,0	Optional	-
cylinder_length	Floating point	> 0,0	Optional	-
head_diameter	Floating point	> 0,0	Optional	-
head_height	Floating point	> 0,0	Optional	-
shear_strength	Floating point	> 0,0	Optional	Dependency from sheet thicknesses
peel_strength	Floating point	> 0,0	Optional	Dependency from sheet thicknesses
material	Alphanumeric	Alphanumeric	Optional	-
part_code	Alphanumeric	Alphanumeric	Optional	-



Key

1 head_height
2 head_diameter
3 length
4 cylinder_length
5 shaft_diameter
a Examples of different nail types.

Figure 39 — Key measures of a nail and examples of different nail types

The following list explains the attributes, see also Figure 39:

— `nail_type`: the alphanumeric name of the nail (naming convention based on supplier nail codes). For more details see Reference [13].

— `shaft_diameter`: the diameter of the shaft of the (unmounted) nail,

— `length`: the overall length of the nail,

— `cylinder_length`: the length of the cylindrical part of the nail shaft

— `head_diameter`: the diameter of the head of the nail,

— `head_height`: the height of the nail head,

— `shear_strength`: Shear failure where the joint fails by shearing a hole in the cover part side material. It is defined as maximum measured force during the test process,

— `peel_strength`: Pull failure in peeling test is where the joint, that is nail and cover sheet, pull apart leaving the base sheet part. It is defined as maximum measured force during the test process,

— `material`: the material of the nail,

— `part_code`: the part code of the nail, as used for example in a PDM system. It can be convenient to use the nail norm (according to e.g. ISO, EN, BSW, DIN) as part code.

There is no “base” attribute for nails since this information can be derived from connection direction.

If possible, a `<nail/>` should know the direction of fixation, therefore, possess a nested element `<normal_direction/>`. However, this is not mandatory in order to allow for importing incomplete data. The direction sense of `<normal_direction/>` is from nail head to tip. The direction element definition can be found in 9.1.3.

The element `<nail/>` allows for following nested elements (see Table 75):

Table 75 — Nested elements of element `<nail/>`

Nested elements	Multiplicity	Use	Constraint
normal_direction	1	Optional	-
tangential_direction	1	Optional	-

EXAMPLE

```
  <connection_0d label="NAIL_100">
      <nail shaft_diameter="10.0" length="26.0" head_diameter="15.0" material="steel" shear_
strength="5200" peel_strength="5000">
          <normal_direction x="0" y="0" z="-10"/>
      </nail>
      <loc> 1645.83 821.145 616.585 </loc>
      <appdata>
          ...
      </appdata>
  </connection_0d>
```

9.11 Rotation joints

9.11.1 General

A rotation joint is denoted by an element `<rotation_joint/>`. This element is described completely by its attributes and nested elements (see Table 76).

Table 76 — Nested elements of `<connection_0d/>` **for** `<rotation_joint/>`

Nested elements	Multiplicity	Use	Constraint / Remarks
rotation_joint	1	Optional	-
Loc	1	Required	-
Appdata	1	Optional	See 7.3.2.
Femdata	1	Optional	See 7.3.3.
custom_attributes_list	1	Optional	See 8.5.

The XML specification of the `<rotation_joint/>` element is shown in Table 77:

Table 77 — Attributes of element `<rotation_joint/>`

Attributes	Type	Value space	Use	Constraint / Remarks
Diameter	Floating point	> 0,0	Optional	-

The following list explains the attribute:

— `diameter`: the diameter of the shaft of the rotation joint.

If possible, a rotation joint should know the direction of fixation, therefore, possess a nested element `<normal_direction/>`. However, this is not mandatory in order to allow for importing incomplete data. The direction sense of `<normal_direction/>` is from the joint's head to point, which element's definition can be found in 9.1.3.

Specific subtypes of rotation joints are defined by adding related nested elements, listed in Table 78:

Table 78 — Nested elements of element `<rotation_joint/>`

Nested elements	Multiplicity	Use	Constraint / Remarks
normal_direction	1	Optional	-
tangential_direction	1	Optional	-
Rotav	1	Required	

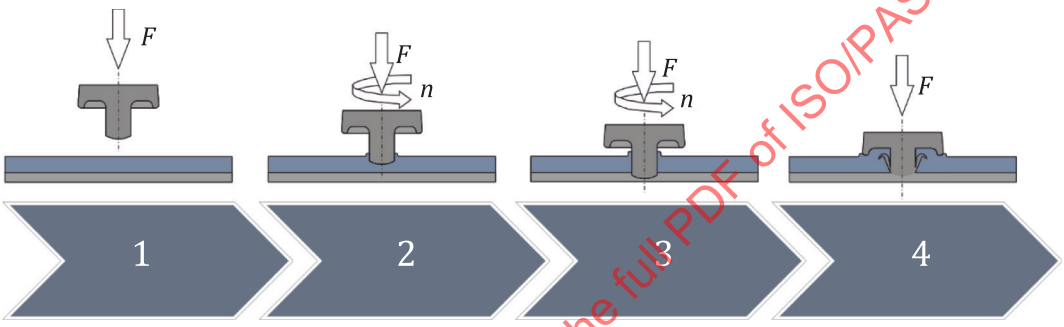
The subtypes are described in detail in the following subclauses.

EXAMPLE

```
<connection_0d label="RJ_2123921">
    ...
    <rotation_joint diameter="3.0">
        <normal_direction x="0" y="0" z="3"/>
        <rotav/>
    </rotation_joint>
    <loc> 1645.83 821.145 616.585 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

9.11.2 ROTAV

ROTAVs are suitable for steel-aluminium connections. Connections of two or three sheets are possible. High grade steel sheets can be used. A description of this technology can be found in Reference [14]. Figure 40 sketches the manufacturing process. Figure 41 depicts a microsection.



Key

1 finding
2 penetrating
3 shaping
4 welding

Figure 40 — Process of rotation joining (ROTAV)[14]

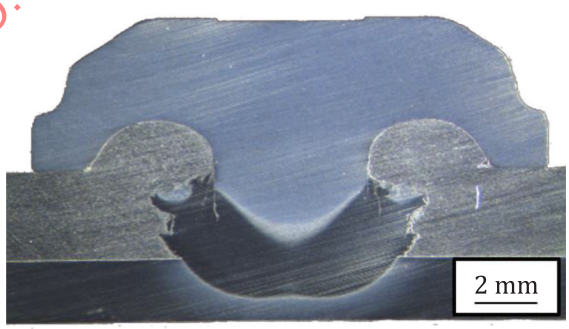


Figure 41 — ROTAV connecting aluminium and steel sheets[14]

The basic steps in the ROTAV process are:

a) Applying rotational velocity and pressure to the ROTAV plug,

b) ROTAV plug penetrating the soft aluminium sheet,

c) ROTAV plug heating base sheet metal (or without pre-punching, both sheet components) and melting through it,

d) Applying compression to the ROTAV arrangement to finish the desired connection.

A ROTAV connection is denoted by an element `<rotav/>`.

For the `<rotav/>` element, the following attributes can be specified (see Table 79):

Table 79 — Attributes of element `<rotav/>`

Attributes	Type	Value space	Use	Constraint
rotational_speed	Floating point	≥ 0,0	Optional	-
compression_force	Floating point	≥ 0,0	Optional	-

The following list explains the attributes:

— `rotational_speed`: In order to facilitate the penetration in the metal sheet of the tip of the ROTAV, it is rotated at a high speed;

— `compression_force`: In order to achieve the fastening properties, the ROTAV is compressed with a vertical force.

The element `<rotav/>` does not allow for any nested elements.

EXAMPLE 1 Minimum definition of a “Rotav” connection

```
<connection_0d label="ROTAV_96930">
    <rotation_joint>
        <rotav/>
    </rotation_joint>
    <loc> 1500.3809 838.75885 730.6529 </loc>
</connection_0d>
```

EXAMPLE 2 Maximum definition of a “Rotav” connection

```
<connection_0d label="ROTAV_96930">
    <rotation_joint diameter="4.0">
        <rotav rotational_speed="1500000" compression_force="10000"/>
        <normal_direction x="0" y="0" z="-10"/>
    </rotation_joint>
    <loc> 1500.3809 838.75885 730.6529 </loc>
    <appdata>
        ...
    </appdata>
</connection_0d>
```

10 1D connections

10.1 Generic definitions

10.1.1 Identification

For identifying 1D connections, the same rules apply as for 0D connections, see 9.1.1.

10.1.2 Location

10.1.2.1 General

The definition of the connection line is one or multiple polylines (sections). Each of the polylines is described as a series of points (vertices). All other curves can also be represented with this type of representation by adding necessary points and thus approximating to the needed accuracy.

The polylines do not need be joined to each other. This is to simulate gaps along the application of a seam or an adhesive, due to crossing another weld, or an obstacle, e.g. a hole in the connected sheets.

At any inner point of a polyline, any kink angle can occur, in principle. However, CAD systems typically do not generate individual curves with kink angles that deviate significantly from the straight line. They retain the GC1 continuity. Therefore, a connection line can be represented by several CAD curves and thus give raise to several polylines. This yields another reason for splitting the connecting lines into several polylines.

χMCF specifies the order of polylines, as well as the order of the locations within each individual polyline.

10.1.2.2 Element <loc_list/>

The list of locations for the definition of the connection line is stored in the element <loc_list>. This element contains nested elements <loc/> defining the location of a point of the connection line in space. These locations have to be ordered so that the line defined by the ordered list of locations specifies the connection line.

The attributes associated to the element <loc_list/> are (see Table 80):

Table 80 — Attributes of element <loc_list/>

Attributes	Type	Use	Constraint
index	Integer	Optional	Required only if there are more than one <loc_list/> elements in the <connection_id/>.

If connection line is made of several polylines, this is expressed by a series of <loc_list/> elements. In this case, the <loc_list/> order is indicated by the index attribute.

The <loc_list/> element has the following nested elements (see Table 81):

Table 81 — Nested elements of <loc_list/>

Nested elements	Multiplicity	Use	Constraint
Loc	1-*	Required	-

10.1.2.3 Element <loc/>

Each location specified by the element <loc/> contains three values specifying the x, y, and z coordinates of the location.

The attributes associated to the element <loc/> are (see Table 82):

Table 82 — Attributes of element <loc/>

Attributes	Type	Use	Constraint
V	Floating point	Required	-

The attribute v is used as surrogate index to ensure proper ordering. The values are not related to the attribute u used in the <weld_position/> element.

The <loc/> with the minimum value of "v" marks the start of a seam weld and *max(v)* is used to mark the end. The reason for this is that some manufacturing techniques are not "symmetric" regarding both ends of a connection line.

EXAMPLE 1 Connection line with a single polyline/section:

```
<loc_list>
      <loc v="0"   > 2581.21 -708.408 31.6532 </loc> <!-- first point -->
      <loc v="0.1" > 2581.42 -708.357 35.2816 </loc>
      <loc v="2.22">  2581.05 -708.302 39.0643 </loc> <!-- last point -->
</loc_list>
```

EXAMPLE 2 A connection line consisting of two disjoint polylines/sections:

```
<loc_list index="1"> <!-- first section -->
      <loc v="0"   > 2581.21 -708.408 31.6532 </loc> <!-- first point -->
      <loc v="1"   > 2581.42 -708.357 35.2816 </loc>
      <loc v="2.22"> 2581.05 -708.302 39.0643 </loc> <!-- last point -->
</loc_list>
<loc_list index="2"> <!-- second section -->
      <loc v="1"  s> 2581.05 -708.302 40.3340 </loc>  <!-- first point -->
      <loc v="2.1"> 2581.05 -708.302 48.5300 </loc>  <!-- last point -->
</loc_list>
```

10.1.3 Intermittent connection lines

10.1.3.1 General

Intermittent connection lines are connection lines, which are fixed only at certain segments along their total length. The gaps between the segments are called “spacings” to avoid confusion with the gap between the connected parts. The benefit of intermittent connection lines compared with individual connection lines is the reduction of administrative overhead.

Intermittent connection lines were introduced with χMCF version 3.1.1 and are only applicable to seam welds, currently.

The total length L_{total} of a connection line is the length of the `<loc_list/>` polygon. Therefore, the total length contains the lengths of both, the segments and the spacings between, before and after segments.

The `<loc_list/>` polygon only approximates exact geometry. This can lead to unavoidable deviations between the length of both, and therefore, to the exact positions of segments, especially next to the end of a connection line. Thus, the reliable definition of intermittent connection lines requires a certain accuracy of this polygon. Additionally, the parameters describing the segmentation shall be consistent in the sense that the segmentation is feasible both, geometrically and with respect to manufacturing. It is not within the scope of the χMCF format to take these responsibilities since additional external information would be required.

From applications such as durability and fatigue, it is known that the beginnings and ends of a seam weld are most relevant for the durability of the connection. Therefore, it shall be guaranteed as far as possible that there exist complete segments. Ultimately, it is the responsibility of the system that creates the χMCF data that chopped final segments do not occur.

Therefore, the following rules apply:

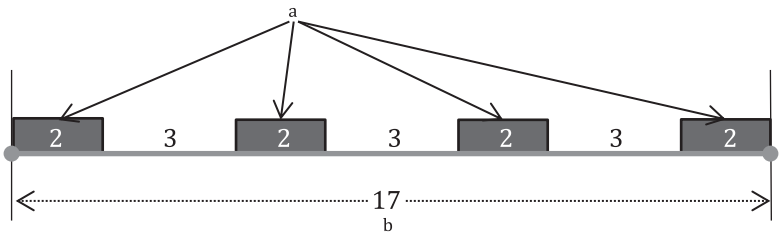
a) Master rule: The creating system alone is responsible for accurate and consistent definition of the segments.

b) If it is required that any segment length (especially first or last) deviates from other segment lengths, a `<segment_list/>` must be used. `<regular_segments/>` are not intended to provide this feature.

c) The excess of segments at the end of a seam weld is not allowed.

10.1.3.2 Terminology

The basic parameters of an intermittent connection line are introduced in Figure 42:



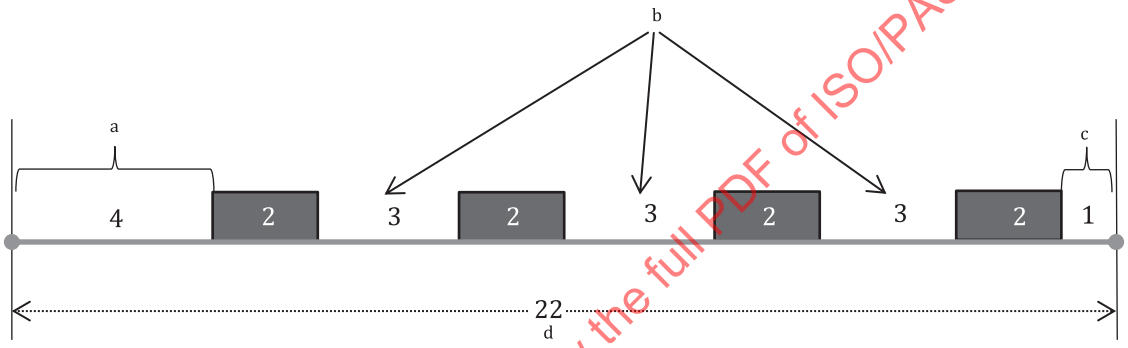
Key

a Segments.

b Total length L_{total} of connection.

Figure 42 — Terminology of a regular intermittent weld

In Figure 42, the connection line has a "total length" L_{total} of 17,0. Its "number of segments" is 4. Each segment is of "length" l = 2,0. The welded segments have a "spacing" of s = 3,0. Note that the first and last segments match the start and end of the connection line.



Key

a First spacing.

b Regular spacing.

c Last spacing.

d Total length L_{total} of connection line.

Figure 43 — Regular intermittent weld with first spacing and last spacing

In Figure 43, the welded segments have a special "first spacing" of 4,0 and a "last spacing" of 1,0, at the beginning and end of the connection line, respectively. Note that "spacing" s is the gap between successive welds, in contrast with the gap at the begin and end of the connection line.

The "density" d of the welded portion of the weld is defined as:

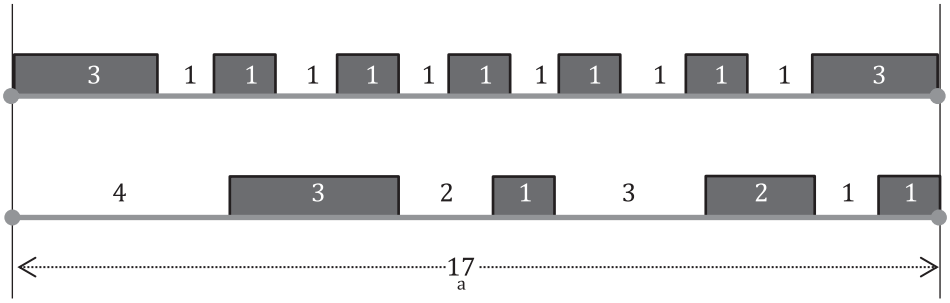
$$d := \frac{l}{l+s}$$

where

l is the length of each segment;

s is the spacing.

For the example above, the density of the welded line is d = 2/5.



Key

[a] Total length L_{total} of connection line.

Figure 44 — Irregular intermittent welds

The intermittent welds in Figure 44 are not regular. Therefore, they are treated as `<segment_list/>`, where each segment is described separately. When all welded segments have the same length and when all gaps between segments have the same spacing, the connection can be represented as sequence of `<regular_segments/>`.

In summary, for the description of an intermittent connection line, the following variants are available:

— `<segment_list/>`: All segments are specified individually with start and end given in curve length parameters of the `<loc_list/>` polygon.

— `<regular_segments/>`: All segments have identical length. All spacings have identical length except for a first spacing at the beginning of the `<loc_list/>` polygon (i.e. before the first segment) and a last spacing at the end of the `<loc_list/>` polygon (i.e. after the last segment).

The elements `<segment_list/>` and `<regular_segments/>` can only be used mutually exclusively. That means, only one of these elements may occur in one `<weld_position/>` element.

The XML specification of the `<segment_list/>` element states:

The `<segment_list/>` element does not have any attributes, but at least one nested element `<segment/>`.

The XML specification of the `<segment/>` element (with L_{total} ≔ total length of the `<loc_list/>` polygon) is shown in Table 83:

Table 83 — Attributes of element `<segment/>`

Attributes	Type	Value space	Use	Constraint
from	Floating point	≥ 0	Required	from < to
to	Floating point	> 0	Required	

If there is more than one segment in the `<segment_list/>`, it is required that all segments s_n can be arranged in a way that $s_{n,\,to} < s_{n+1,\,from}$ is true for *n=1, ..., num_segments - 1*.

The XML specification of the `<regular_segments/>` element (with L_{total} ≔ length of the `<loc_list/>` polygon and n ≔ number of segments, both positive) is shown in Table 84:

Table 84 — Attributes of element `<regular_segments/>`

Attributes	Type	Value space	Use	Constraint
num_segments	Integer	> 0	Required	
Length	Floating point	> 0	Required	
Spacing	Floating point	> 0	Required	
first_spacing	Floating point	≥ 0,0 (default: 0)	Optional	
last_spacing	Floating point	≥ 0,0 (default: 0)	Optional	
Keep	Selection	spacing, length, density (default)	Optional	
max_percentage_of_compensation	Floating point	> 0,0 and ≤ 100,0 (default: 1,0)	Optional	If both attributes are missing, default of "max_percentage_of_compensation" is used. Only one of "max_absolute_compensation" or "max_percentage_of_compensation" may be specified.
max_absolute_compensation	Floating point	≥ 0,0	Optional	

The description of `<regular_segments/>` requires parameters with specific semantics, as listed in Table 84:

— `num_segments`: Prescribed number of welded segments,

— `length`: Prescribed length of every segment,

— `spacing`: Prescribed length of any inner spacing, a spacing between two segments,

— `first_spacing`: Length of the spacing before the first segment, if any, (default: 0)

— `last_spacing`: Length of the spacing after the last segment, if any, (default: 0)

— `keep`: Strategy about how to cope with the case that all prescribed segments and spacings together differ from the total length of the `<loc_list/>` polygon,

— `max_percentage_of_compensation`: The maximum allowable deviation, as a percentage, of the resulting size of length or spacing over its prescribed size. A warning has to be issued, if the adjusted value deviates from the prescribed value by more than `max_percentage_of_compensation`. Valid range is from 0,0 to 100,0 %.

— `max_absolute_compensation`: The maximum allowed deviation, in length units, of the difference between the resulting size of length or spacing and its prescribed size. A warning has to be issued, if the adjusted value deviates from the prescribed value by more than `max_absolute_compensation`.

Semantics of the different possible values of `keep` parameter:

— `spacing`: Spacing between segments is kept. Length is adjusted,

— `length`: Segment lengths are kept. Spacing between segments is adjusted,

— `density`: Effective density *d* is kept. This implies that both, segment lengths and spacing absorb the change proportionally, but `first_spacing` and `last_spacing` remain unchanged.

In all cases, the number of segments is kept unchanged.

10.1.3.3 Formulae for adjusting the segment sizes according to the total length of the connection line

According to Figure 43, the welded segments in a connection line are spread over the effective welded length L_{eff} between the first and the last spacing. The size of $L_{effective}$ is given by (further details are given in Annex A):

$$L_{eff} = L_{total} - m_{first} - m_{last}$$

where

L_{total} is the total length of the polyline;

L_{eff} Is the effective welded length;

m_{first} is first_spacing;

m_{last} is last_spacing.

The number of segments n is given by attribute `num_segments`.

NOTE The number of spacings is always $n–1$.

— When keep = "`length`", the adjusted spacing is calculated with this formula:

$$\overline{s} := \frac{L_{eff} - nl}{n-1}$$

— When keep = "`spacing`", the adjusted length is calculated with this formula:

$$\overline{l} := \frac{L_{eff} - (n-1)s}{n}$$

— When keep = "`density`", the adjusted length and adjusted spacing are given by these formulae:

$$\overline{l} := \frac{dL_{eff}}{n-1+d}$$

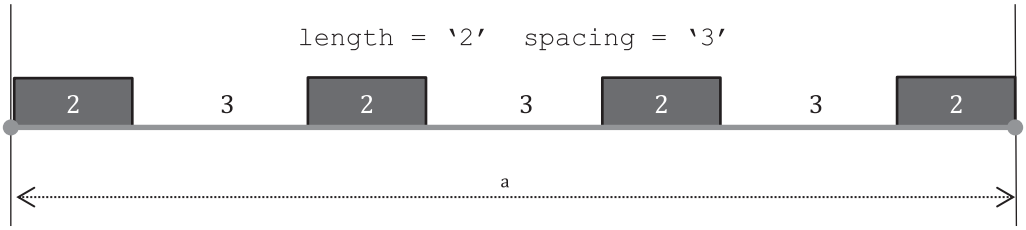
and

$$\overline{s} := \frac{(1-d)L_{eff}}{n-1+d}$$

where d is the prescribed density, calculated by

$$d := \frac{l}{l+s}$$

EXAMPLE 1 `<corner_weld/>` with `<regular_segments/>` and "Required" attributes only (see Figure 45).



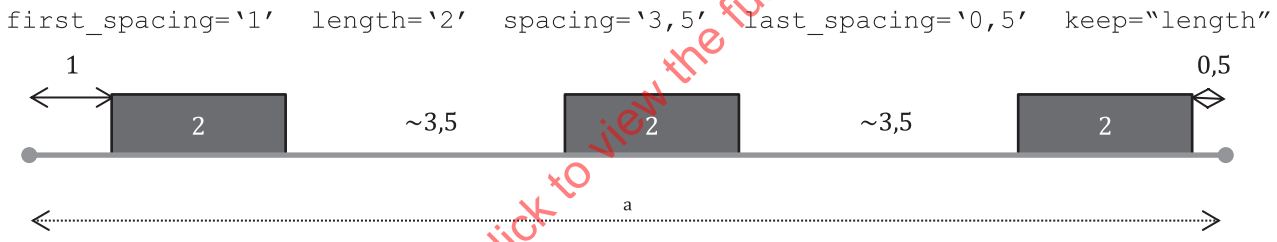
Key

a L_{total} = 17.

Figure 45 — `<corner_weld/>` with `<regular_segments/>` and "Required" attributes only

```
<seamweld>
    <corner_weld base="1" technology="resistance">
        <weld_position u="0.2" x="1" y="0" z="1">
            <regular_segments num_segments="4" length="2" spacing="3"/>
        </weld_position>
        <sheet_parameter ... />
    </corner_weld>
</seamweld>
```

EXAMPLE 2 Regular single sided welding (a `<corner_weld/>` with `<regular_segments/>` and all attributes, see Figure 46)



Key

a L_{total} = 14,435.

Figure 46 — Regular single sided welding (a `<corner_weld/>` with `<regular_segments/>` and all attributes)

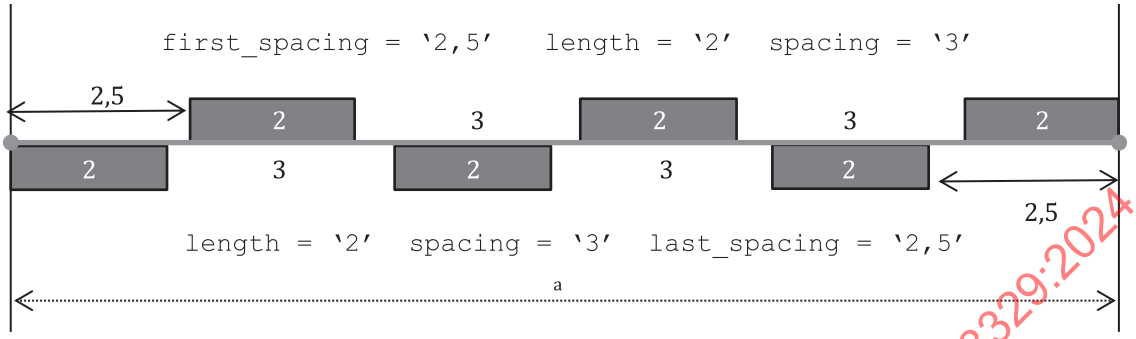
```
<seamweld>
    <corner_weld base="1" technology="resistance">
        <weld_position u="0.2" x="1" y="0" z="1">
            <regular_segments
                num_segments="3"
                first_spacing="1.0" last_spacing="0.5" length="2.0" spacing="3.5"
                keep="length" max_absolute_compensation="0.2"/>
        </weld_position>
        <sheet_parameter ... />
    </corner_weld>
</seamweld>

...

<seamweld>
    <corner_weld base="1" technology="resistance">
        <weld_position u="0.2" x="1" y="0" z="1">
            <regular_segments
                num_segments="3"
                first_spacing="1.0" last_spacing="0.5" length="2.0" spacing="3.5"
                keep="length" max_percentage_of_compensation="3.0"/>
```

```
        </weld_position>
        <sheet_parameter ... />
    </corner_weld>
</seamweld>
```

EXAMPLE 3 Staggered welding (a <corner_weld/> welded from both sides in alternating sequence, with two <regular_segments/> for the two <weld_position/>s, see Figure 47)



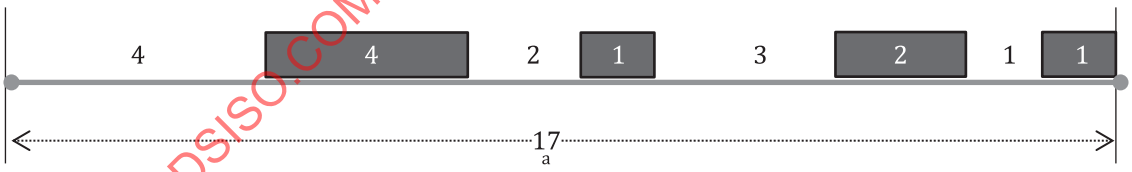
Key

a L_{total} = 14,5.

Figure 47 — Staggered welding

```
<seamweld>
    <corner_weld base="1" technology="resistance">
        <weld_position u="0.2" x="1" y="0" z="1">
            <regular_segments
                first_spacing="2.5" num_segments="3" spacing="3.0" length="2.0"/>
        </weld_position>
        <weld_position u="0.5" x="-1" y="0" z="1">
            <regular_segments
                last_spacing="2.5" num_segments="3" spacing="3.0" length="2.0"/>
        </weld_position>
        <sheet_parameter ... />
    </corner_weld>
</seamweld>
```

EXAMPLE 4 Definition of a <corner_weld/> with <segment_list/> (see Figure 48).



Key

a L_{total} = Total length of connection line.

Figure 48 — Definition of a <corner_weld/> with <segment_list/>

```
<seamweld>
    <corner_weld base="1" technology="resistance">
        <weld_position u="0.2" x="1" y="0" z="1">
            <segment_list>
                <segment from="4.0" to="7.0" />
                <segment from="9.0" to="10.0" />
                <segment from="13.0" to="15.0" />
                <segment from="16.0" to="17.0" />
            </segment_list>
        </weld_position>
        <sheet_parameter ... />
    </corner_weld>
```

```
</seamweld>
```

NOTE The order of `<segment/>` lines is arbitrary since segments are not allowed to overlap.

10.1.4 Type specification

Each connection should be assigned a type during its life cycle. The XML definitions of all 1D connections contain the following elements (see Table 85):

Table 85 — Nested elements of element `<connection_1d/>`

Nested elements	Multiplicity	Use	Constraint / Remarks
seamweld	1	Optional	-
adhesive_line	1	Optional	-
hemming	1	Optional	-
sequence_connection_0d	1	Optional	-
contact_list	1	Optional	See 7.4.3.6.
stacking	1	Optional	See 7.4.2.4.

Up to one of the type elements (`seamweld`, `adhesive_line`, `sequence_connection_0d`, `hemming`) may exist in a `<connection_1d/>`. If none of the type elements is given, the type defaults to `<seamweld/>`.

10.2 Seam welds

χMCF knows several kinds of 1D connections. Of these, seam welds are addressed first.

10.2.1 Description and modelling parameters

Several cross-section geometries and modelling alternatives are established for seam welds. However, χMCF does not support changing these characteristics in the course of a single seam weld. If necessary, a seam weld must therefore be split into several.

This ensures that a seam weld definition only represents one cross-section with the welding parameters for all the welded sides.

NOTE Several welding technologies produce material structures which are oriented. In particular, there is a difference between the start and the end of a seam weld. χMCF knows about the orientation of a seam weld and therefore, it can distinguish between start and end. But it does not yet provide means to transport details about the difference between both, neither for CAE nor CAM.

10.2.2 Seam weld definition overview

The weld definition depends on the type of the seam weld. The parameters and their meaning can be different for each of the different seam weld types. Detailed descriptions are provided in the next clauses, where each weld type is described separately.

Figure 49 provides an overview of the currently supported seam weld types and their parameters.

For each of the seam weld type, it contains the following information:

— type of the seam weld,

— number of weld positions for the type,

— supported technologies,

— widely used weld sections for the respective weld type (other sections are generally permitted by the standard, but feasibility and compatibility must be ensured by the designer.),

— required parameters,

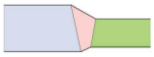
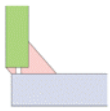
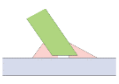
— optional parameters with their default values,

— section drawing / layout related to the seam weld type.

For the given combinations of seam weld type, technology and cross-section, the parameters and the section drawings are provided. The section drawings do not show the specific sections possible for a technology.

Parameters describing sheet thicknesses are not part of the χMCF file contents. They need to be derived from CAD geometry or FE meshes. However, they are used in the seam weld type specific sections to describe parameters contained in the χMCF file and their relations.

The variety of seam weld types is to be handled by the application. The content of χMCF was selected so that, in combination with information from geometry or meshes, the specific type of a connection can be determined.

Weld type	# Weld positions	Welding technology	Section	Weld parameter			Layout
Butt weld	1		I	width	-	-	
			V	width	-	-	
			U	width	-	-	
			X	width	-	-	
			Y	width	-	-	
			Radius		-	-	
Corner weld	1-2	Fillet	Fillet	thickness	penetration=0, gap=0, angle=45	-	
			HV	thickness	gap=0, angle=45	penetration=1	
			U	thickness	penetration=0, gap=0, angle=45		
	2-4	Fillet	Fillet	thickness	penetration=0, gap=0, angle=45	-	
			HV	thickness	gap=0, angle=45	penetration=1	
			U	thickness	penetration=0, gap=0, angle=45		
Edge weld	1		I	width	gap=0	-	
	1		V			-	
	1		U			-	
I-weld	1	Laser	-	width	gap=0	-	
	1	Fillet	U				
Overlap weld	1	Fillet	-	thickness	penetration=0, gap=0, angle=45	-	
	2	Fillet	-	thickness	penetration=0, gap=0, angle=45	-	
	2	Fillet	-	thickness	penetration=0, gap=0, angle=45	-	
Y-joint	1-2	Fillet	Fillet	thickness	penetration=0, gap=0, angle=45	-	
			HV	thickness	gap=0, angle=45	penetration=1	
			HY	thickness	penetration=0, gap=0, angle=45	-	
K-joint	2-3	Fillet	Fillet	thickness	penetration=0, gap=0, angle=45	-	
			HV	thickness	gap=0, angle=45	penetration=1	
			HY	thickness	penetration=0, gap=0, angle=45	-	

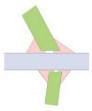
Weld type	# Weld positions	Welding technology	Section	Weld parameter			Layout
Cross-joint	2-4	Fillet	Fillet	thickness	penetration=0, gap=0, angle=45	-	
			HV	thickness	gap=0, angle=45	penetration=1	
			HY	thickness	penetration=0, gap=0, angle=45	-	

Figure 49 — Seam weld types and attributes

10.2.3 Specific XML realization

This part of the XML structure describes the data that is stored for each of the seam welds. This includes the necessary details to describe each connection in depth.

Within the XML definition of a seam weld, each of the associated physical welds is stored in a separate weld position in the specific subtype definition. Figure 50 contains an example:

```
<connection_list>
  <connection_1d label="1000032">
    <loc_list>
      <loc v="0">68 0 0</loc>
      <loc v="1">88 0 0</loc>
    </loc_list>
    <seamweld>
      <butt_joint base="1" technology="resistance" section="Y" filler="yes">
        <weld_position u="1" x="0" y="6.12323e-17" z="1" width="3" />
      </butt_joint>
    </seamweld>
    <appdata>
      <MEDINA xmlns="http://servicenet.t-systems.com/medina/xMCF">
        <data_at_connector>
          <original_loc_list>
            <loc u="0">68 4 10</loc>
            <loc u="1">88 4 10</loc>
          </original_loc_list>
          <connection_data adjust_limit="1" weld_position_id="1" max_projection_distance="10" />
          <administrative_data element_label="Weldline_Overlap-Join"
           part_tree_position="fixed" connector_property_id="1000032" />
        </data_at_connector>
      </MEDINA>
    </appdata>
  </connection_1d>
</connection_list>
```

Figure 50 — χMCF structure of a seam weld (connection_1d)

10.2.4 Generic seam weld definition

10.2.4.1 Type specification

Each seam weld is characterized by its main type. It is described more precisely by its subtype. This means there is a general category that includes several subcases. Detailed information is given in the following 10.2.4.1.1 and 10.2.4.1.2.

10.2.4.1.1 Definition of main type

The element main type for a seam weld is always `<seamweld/>`. This element is located directly below the `<connection_1d/>` element. It is used to define the connection as general as it can be.

The XML definition of seam weld main type contains the following nested elements (see Table 86):

Table 86 — Nested elements of element `<seamweld/>`

Nested elements	Multiplicity	Use	Constraint
butt_joint	1	Optional	-
corner_weld	1	Optional	-
edge_weld	1	Optional	-
i_weld	1	Optional	-
overlap_weld	1	Optional	-
y_joint	1	Optional	-
k_joint	1	Optional	-
cruciform_joint	1	Optional	-
flared_joint	1	Optional	-

EXAMPLE Main type `<seamweld/>`

```
<connection_1d>
    <seamweld>
        ...
    </seamweld>
</connection_1d>
```

NOTE The differentiator for the specific seam welds is stored as value in the subtype element which is described below.

10.2.4.1.2 Definition of subtype

Different kinds of welds are distinguished through the definition of a subtype of the seam weld.

Valid values for the subtype element are:

— `butt_joint`,

— `corner_weld`,

— `edge_weld`,

— `i_weld` (not be confused with cross-section “I”, as described in 10.2.4.4.7),

— `overlap_weld`,

— `y_joint` (not be confused with cross-section “Y”, as described in 10.2.4.4.11),

— `k_joint`,

— `cruciform_joint`,

— `flared_joint`.

Each subtype element can contain the following attributes (see Table 87):

Table 87 — Attributes of element `<subtype/>`

Attributes	Type	Value space	Use	Constraint
base	Integer	> 0	Optional	-
technology	Selection	resistance arc laser friction brazing	Optional	-

Each subtype element contains the following nested elements (see Table 88):

Table 88 — Nested elements of element `<subtype/>`

Nested elements	Multiplicity	Use	Constraint
weld_position	1 - *	Optional	-
sheet_parameter	1 - *	Optional	-

NOTE The number of elements of `<weld_position/>` is dependent on the specific subtype.

10.2.4.1.3 Attribute "base"

The attribute `base` defines the index of the base sheet for the seam weld. It references the attribute `index` inside the element `<part/>` of the `<connected_to/>` element. This is especially useful when the angle of the weld itself is not symmetrical between the welded sheet and the base sheet. That means it is crucial to precisely specify to which sheet part the angle is measured.

10.2.4.1.4 Attribute "technology"

The technology used to weld the connection can be specified for each of the welds of a connection separately.

This technology can be one of

— resistance welding,

— arc welding,

— energy beam welding (laser, for example),

— friction welding,

— brazing (not allowed for I-welds, for technical reasons).

In addition to the technology, there is a specification for each of the weld positions whether the welding introduces additional material (attribute `filler`).

The attribute `technology` defines the welding technology used for its subtype.

Possible values are:

— `resistance`,

— `arc`,

— `laser` (energy beam / laser),

— `friction`,

— `brazing`.

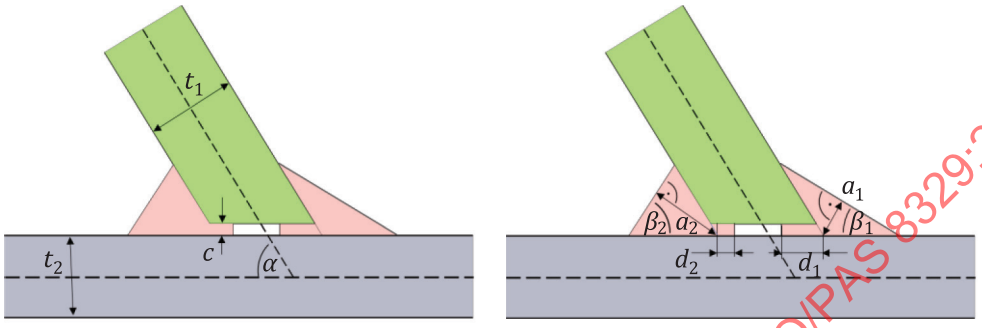
EXAMPLE Main type `<seamweld/>` with `<butt_joint/>` as subtype:

```
<connection_id>
    <seamweld>
        <butt_joint base="1" technology="resistance">
            ...
            <weld_position ... />
            <sheet_parameter ... />
            ...
        </butt_joint>
    </seamweld>
</connection_id>
```

10.2.4.2 Weld position and sheet metal parameters

It is intended to collect the parameters that can be observed in relation to the welding processes in specific elements. Some of the used and measured parameters are related to the involved sheet metal parts, describing the thickness of the sheet and the applied sheet angle between each two sheet metal parts.

On the other hand, the parameters that are mentioned in terms of the welding process related to the weld itself can be distinguished. The detailed description of these parameters can be seen for sheet parameters in 10.2.4.3 and for weld position parameters in 10.2.4.4. Accordingly, each seam weld can have as many weld-position-related parameters (e.g. a-value), as it has weld positions. In Figure 51 for instance its two of both.



Key

t_i	sheet thickness	a_i	weld throat thickness
α	joint angle	β_i	weld angle
c	gap	d_i	depths of the penetrations

Figure 51 — Sheet parameters versus weld position parameters

10.2.4.3 Parameters assigned to a specific sheet of the flange

10.2.4.3.1 General

In a welded connection, there are different kinds of parameters that shall be assigned either to a welded sheet metal or to the created weld itself. Accordingly, those parameters are grouped under two elements, residing directly under the parent subtype element. These are the elements `<sheet_parameter/>` and the `<weld_position/>`.

10.2.4.3.2 Element `<sheet_parameter/>`

The element `<sheet_parameter/>` describes the sheet in order to identify the correct sheet when multiple sheets are connected. Furthermore, it defines as an attribute the corresponding gap applied between the welded sheet and the base sheet, which is in general the applied gap between the welded sheets involved in the welding process.

It is defined using the following attributes (see Table 89):

Table 89 — Attributes of element `<sheet_parameter/>`

Attributes	Type	Use	Constraint / Remarks
index	Integer	Required	It shall be referenced to `<part/>` index attribute
gap	Floating point	Optional	Default value is 0
sheet_thickness	Floating point	Optional	-
sheet_angle	Floating point	Optional	-

10.2.4.3.3 Attribute `index`

The value of the attribute `index` shall be referenced to the part `index`. The index needs to be unique only within the parent element `<connected_to/>`. For specific connections, it is used as the matching index for the subjected welded sheet.

10.2.4.3.4 Attribute `gap`

The value of the attribute `gap` is numerical in the range [0, ∞). It defines the distance between the base and the connected sheet.

10.2.4.3.5 Attribute `sheet_thickness`

The value of the attribute `sheet_thickness` is numerical in the range (0, ∞). It defines the CAD related input for the thickness measure of the connected sheet (in Figure 51, this is t_2). In case more than one welded sheet exists, see the definition in 10.2.11.6.

10.2.4.3.6 Attribute `sheet_angle`

The value of the attribute `sheet_angle` is numerical in the range [0, 360). It describes the angle between the central surfaces of the base sheet and the connected sheet.

EXAMPLE

```
<connection_1d>
    <seamweld>
        <corner_weld base="1" technology="resistance">
            <weld_position .../>
            <sheet_parameter index="2" gap="1.0" sheet_thickness="1.5" sheet_angle="90"/>
        </corner_weld>
    </seamweld>
</connection_1d>
```

10.2.4.4 Welding position

10.2.4.4.1 Basic definitions

The geometric position of the physical welding of the seam weld is specified by an orientation vector pointing from the weld root into the side where the welding takes place (see Figure 52).

The origin of this orientation vector is located directly on the `<loc_list/>` polyline. The position on the `<loc_list/>` polyline is determined by a fraction in the range [0, 1] of the complete line. The fraction is applied to the length of the `<loc_list/>` polyline measured as sum of all segment lengths in space.

A connection can be welded at different positions. Depending on the seam weld type, between two and five positions can occur (e.g. by combining a K-joint with a Y-joint). Each position represents a physical welding performed from one side of the structure.

Details for each seam weld type are described inside the specific section (see subclauses 10.2.5 to the end of 10.2).

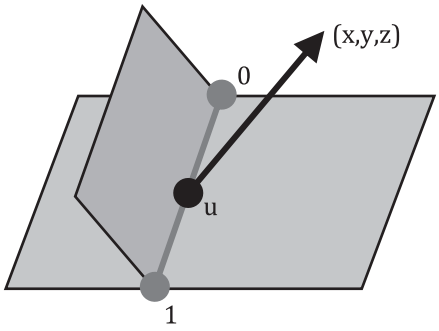


Figure 52 — Welding position of a Y-joint

10.2.4.4.2 Primary and secondary sides

For weld definitions that need a specific side, the orientation vector defines the primary side. All other sides are named secondary sides, not specifying any precedence on them.

10.2.4.4.3 Element `<weld_position/>`

The element `<weld_position/>` describes the location of the weld relative to the connection line specified in `<loc_list/>`.

Each `<weld_position/>` element can contain the following nested elements (see Table 90):

Table 90 — Nested elements of element `<subtype/>`

Nested elements	Multiplicity	Use	Constraint
segment_list	0 - 1	Optional	mutually exclusive – For details, see 10.1.3 Intermittent Connection Lines.
regular_segments	0 - 1	Optional	

The element `<weld_position/>` is defined using the following attributes (see Table 91):

Table 91 — Attributes of element `<weld_position/>`

Attributes	Type	Use	Constraint / Remarks
base	Integer	Optional	Value only for specific weld types
u	Floating point	Required	0 ≤ u ≤ 1
x	Floating point	Required	-
y	Floating point	Required	-
z	Floating point	Required	-
reference	Boolean	Optional	"false"
section	Selection	Optional	-
thickness	Floating point	Optional	Value only for specific weld types
width	Floating point	Optional	Value only for specific weld types
angle	Floating point	Optional	-
filler	Selection	Optional	-
filler_material	Alphanumeric	Optional	-
shape	Selection	Optional	-
penetration	Floating point	Optional	0 ≤ penetration ≤ 1

Depending on the subtype, the attributes of the element `<weld_position/>` are different. Each of the subtypes supports its specific combination of attributes. The detailed description of the specific combination can be found in the according subclauses "Element `<weld_position/>`" below.

EXAMPLE

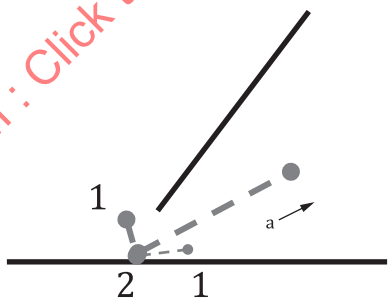
```
<connection_1d>
    <seamweld>
        <corner_weld base="1" technology="resistance">
            <weld_position u="0.2" x="1" y="0" z="1"
                reference="true"
                section="HV"
                thickness="0.5"
                angle="45"
                filler="yes"
                filler_material="E7018-X"
                shape="straight"
                penetration="0.6"/>
            <sheet_parameter index="2" gap="1.0"/>
        </corner_weld>
    </seamweld>
</connection_1d>
```

10.2.4.4.4 Attributes "u", "x", "y", "z"

The attribute `u` specifies the relative location on the `<loc_list/>` polyline. Value `u=0` represents the first location of the connection line matching the element `<loc/>` specified with the lowest value for the attribute `u`. Value `u=1` represents the last location of this line matching the element `<loc/>` with highest value for the attribute value u. Values in between are specifying the point located at the specified fraction of the `<loc_list/>` polyline measure in summed up lengths of the segments of the `<loc_list/>` polyline in space.

The attributes `x`, `y`, and `z` specify the direction vector in the global coordinate system into the quadrant of the welding. The origin of this vector is defined by `u` and the `<loc_list/>` polyline.

The length of the vector has no specific meaning, only the direction is used. However, it should be sufficiently long to be unambiguously visible like illustrated in Figure 53:



Key

1 vague

2 the weld

[a] Good.

Figure 53 — Welding position vector direction and length

10.2.4.4.5 Attribute `reference`

The attribute `reference` specifies this weld position to be the reference for welds that need such a reference. In case of corner welds, butt joints, K-joints or cruciform joints, this is needed to specify a specific side for one of the attributes. For details, see the corresponding subclauses.

10.2.4.4.6 Attribute `section`

The attribute `section` defines the geometry section of the weld. The different section types that can be used inside the definition of seam welds are listed here. The description here denotes the principles of the sections. Details of the interpretation on the different weld type can be found in the corresponding section for each of the weld types.

In most cases the sections “Fillet”, “HV” and “HY” are used in seam weld connections when the head of a sheet is welded on a base sheet. Connections putting two sheet heads together mostly use the section types “I”, “V”, “X” and “Y”.

Widely used values are:

— `I`,

— `V`,

— `U`,

— `X`,

— `Y`,

— `HV`,

— `HY`,

— `Fillet`,

— `Radius`.

10.2.4.4.7 Section “I”

The section “I” describes the filling of the weld normally on the head sides of a connection. The section is filled completely and may be welded from one or two sides.

NOTE Section “I” is not the same as seam weld subtype “`i_weld`” (see 10.2.4.1 Type Specification).

10.2.4.4.8 Section “V”

The section “V” describes the one-sided filling of the weld with welding material that looks like a “V”. The weld filling provides full penetration.

10.2.4.4.9 Section “U”

The section “U” describes the one-sided filling of the weld with welding material that looks like a “U”. The penetration in most cases is less than full penetration.

10.2.4.4.10 Section “X”

The section “X” describes the filling of a two-side weld with welding material that looks like an “X”. The weld filling provides full penetration.

10.2.4.4.11 Section “Y”

The section “Y” describes the one-sided filling of the weld with welding material that looks like a “Y”. Only a part of the gap between the welded sheets is filled, thus there is no full penetration.

NOTE Section “Y” is not the same as seam weld subtype “`y_joint`” (see10.2.4.1 Type Specification).

10.2.4.4.12 Section “HV”

The section “HV” describes the filling of a one-sided weld with a full penetration. The welded sheet has normally to be phased to take full advantage of the full penetration.

10.2.4.4.13 Section “HY”

The section “HY” describes a filling of a one-side weld, but the penetration is only partial. In common cases, the welded sheet is phased partially to take again advantage of the penetration at that area.

10.2.4.4.14 Section “Fillet”

The section “Fillet” describes a one-sided welding placed on the outside of the welded sheets. Depending on the sheet thicknesses, there can be a penetration.

10.2.4.4.15 Section “Radius”

The section “Radius” describes a special case where the welding material looks like a circle but not filling the complete gap between the welded sheets. In most cases there is no full penetration.

10.2.4.4.16 Attribute `thickness`

The value of the attribute `thickness` is a numerical value in the range of (0, ∞). It describes the distance between the weld root and the weld surface. It is used for to describe the throat thickness of the weld.

10.2.4.4.17 Attribute `width`

The value of the attribute `width` is a numerical value in the range of (0, ∞).

10.2.4.4.18 Attribute `angle`

The value of the attribute `angle` is a numerical value. This attribute of the `<weld_position/>` element describes the angle between the weld face and the base sheet face.

10.2.4.4.19 Attribute `filler`

The attribute `filler` specifies whether the welding is performed using filling material. This is the case for resistance or arc welding but not for laser welding.

The allowed values are:

— `yes`

— `no`

According to the above rule on filling material, the default values are depending on the attribute value of `technology` of the element subtype (see Table 92):

Table 92 — Default values of attribute `filler`, dependent from attribute `technology`

Attribute value `technology`	Default value `filler`
resistance	Yes
arc	Yes
laser	No

10.2.4.4.20 Attribute `filler_material`

The attribute `filler_material` specifies the applied material during the welding process.

10.2.4.4.21 Attribute `shape`

The attribute `shape` defines the shape of the weld throat. Allowed values are:

— `straight`,

— `convex`,

— `concave`.

Independent of the shape, the weld position attributes (e.g. a-value/weld throat thickness, weld angle) are taken with respect to the straight line (according to `shape=straight`). The shape value is just a hint to a specific solver. It does not provide an exact definition whether `convex` or `concave` mean, e.g. "a segment of a circle", "a parabolic segment", nor how big the deviation from the straight shape is.

10.2.4.4.22 Attribute `penetration`

The value of the attribute `penetration` is a numerical value in the range [0; 1]. The value describes the ratio between the thickness and the penetration of the sheets. Value of 0 means no penetration, value of 1 represents complete penetration.

NOTE The attribute `penetration` of a `<weld_position/>` holds for all sheets connected by this `<weld_position/>` (e.g. for K-joints). If all `<weld_position/>` at the same welded sheet have a sum of penetration ≥ 1, there is no open (unfilled) gap between the base sheet and the welded sheet.

10.2.5 Butt joint

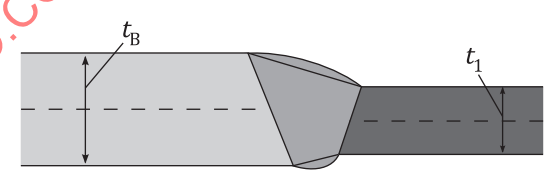
10.2.5.1 General

The principles of the modelling of butt joints for χMCF are described in this subclause. A butt joint describes a connection between two sheets welded at their forehead side.

The XML definition of a butt joint supports up to two weld positions. Each of the weld positions is specified using the element `<weld_position/>` with the corresponding attributes and nested elements inside the subtype definition.

10.2.5.2 Sheet parameters

The parameters to describe the connection are (see Figure 54):



Key

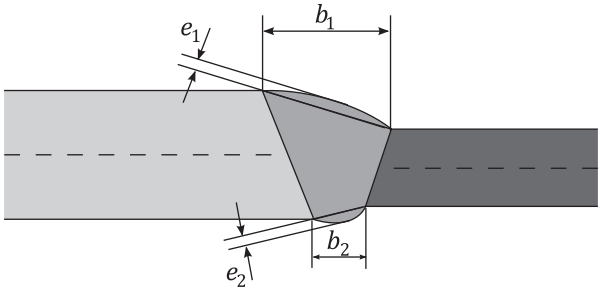
t_B thickness of base sheet

t_1 thickness of welded sheet

Figure 54 — Butt joint sheet layout

10.2.5.3 Weld parameters

The parameters of the weld are described below (see Figure 55):



Key

b_1 width of the weld at primary side,
b_2 width of the weld at secondary side
e_1 reinforcement of the weld at primary side
e_2 reinforcement of the weld at secondary side

Figure 55 — Butt joint parameters

In the χMCF file, the following parameters can be specified (see Table 93):

Table 93 — Parameters of butt joint per `<weld_position/>` (w.p.)

Parameter	χMCF-Key	Multiplicity per w.p.	Value space	Use	Default value
B	width	1	≥ 0	Optional	-
E	-	1	≥ 0	Optional	0

NOTE The reinforcement is not yet supported as an attribute in χMCF version 3.1.1 and earlier.

10.2.5.4 Attributes

10.2.5.4.1 Attribute `base`

The index for the base sheet is specified using the attribute `base`.

10.2.5.4.2 Attribute `technology`

The following list explains the attributes. The value for the attribute `technology` can be specified using the following values:

— `resistance`;

— `arc`;

— `laser` (energy beam / laser);

— `friction`;

— `brazing`.

10.2.5.5 Element `<weld_position/>`

For the element `<weld_position/>`, the following attributes can be specified for the butt joint (see Table 94):

Table 94 — Attributes of element `<weld_position/>` for butt joint

Attributes	Type	Use	Constraint
u	Floating point	Required	0 ≤ u ≤ 1
x	Floating point	Required	-
y	Floating point	Required	-
z	Floating point	Required	-
reference	Boolean	Optional	"false"
section	Selection	Optional	-
width	Floating point	Optional	-
filler	Selection	Optional	-
filler_material	Alphanumeric	Optional	-

10.2.5.5.1 Attributes `u, x, y, z,` and `reference`

The detailed definition is provided in 10.2.4.4.

10.2.5.5.2 Attribute `section`

Valid values for the attribute `section` of a butt joint are:

— `I` (not be confused with seam weld subtype "`i_weld`", see 10.2.4.1),

— `U`,

— `V`,

— `X`,

— `Y` (not be confused with seam weld subtype "`y_joint`", see 10.2.4.1) ,

— `Radius`.

10.2.5.5.3 Attribute `width`

The attribute value `width` specifies the width of the weld.

10.2.5.5.4 Attribute `filler`

Valid values for the attribute filler can be:

— `yes`,

— `no`.

Depending on the technology, the default value can differ, see 10.2.4.4.19 Attribute `filler`.

10.2.5.5.5 Attribute `filler_material`

The attribute `filler_material` specifies the applied material during the welding process.

EXAMPLE 1 `<weld_position/>` with required attributes only

```
<seamweld>
    <butt_joint base="1" technology="arc">
        <weld_position u="0.2" x="1" y="0" z="1"/>
        <sheet_parameter ... />
    </butt_joint>
</seamweld>
```

EXAMPLE 2 `<weld_position/>` with all attributes

```
<seamweld>
    <butt_joint base="1" technology="arc">
        <weld_position u="0.2" x="1" y="0" z="1"
                       reference="true"
                       section="X"
                       width="1.5"
                       filler="yes"
                       filler_material="E7018-X"/>
        <sheet_parameter ... />
    </butt_joint>
</seamweld>
```

10.2.5.6 Element `<sheet_parameter/>`

For the element `<sheet_parameter/>`, the following attributes can be specified for the butt joint (see Table 95):

Table 95 — Attributes of element `<sheet_parameter/>` for butt joint

Attributes	Type	Use	Constraint / Remarks
index	Integer	Required	It shall be referenced to `<part/>` index attribute.
gap	Floating point	Optional	Default value is 0.
sheet_thickness	Floating point	Optional	-
sheet_angle	Floating point	Optional	-

EXAMPLE `<sheet_parameter/>` with all attributes

```
<seamweld>
    <butt_joint base="1" technology="arc">
        <weld_position u="0.2" x="1" y="0" z="1" ... />
        <sheet_parameter index="2" gap="0" sheet_thickness="1.5" sheet_angle="180" />
    </butt_joint>
</seamweld>
```

10.2.6 Corner weld

10.2.6.1 General

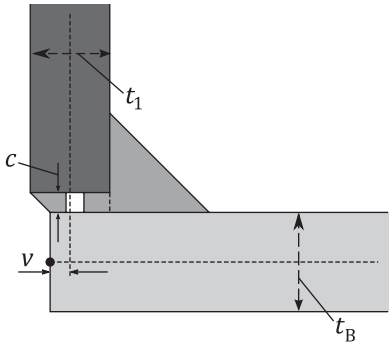
The principles of the modelling of corner welds for χMCF are described in this subclause. A corner weld describes a connection between two or three sheets welded together.

The XML definition of a corner weld supports up to four physical weld positions. Each of the weld positions is specified using the element `<weld_position/>` with the corresponding attributes and nested elements inside the subtype definition.

10.2.6.2 Simple corner weld

10.2.6.2.1 Sheet parameters

The parameters to describe the connection are (see Figure 56):



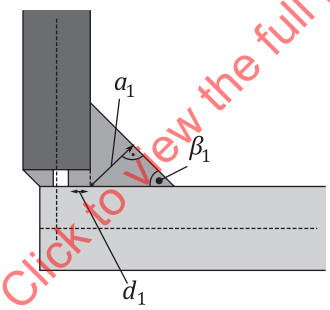
Key

t_B thickness of base sheet
t_1 thickness of welded sheet
c gap between base sheet and welded sheet
v misalignment of welded sheet

Figure 56 — Corner weld sheet layout

10.2.6.2.2 Weld parameters

The parameters of the welds are the same for all the potential physical welds on the connection (see Figure 57):



Key

a_1 thickness of the weld (a-value, throat)
d_1 depth of the penetration
β_1 weld angle

Figure 57 — Corner weld parameters

For the penetration, the ratio η_i of the penetration depth to the sheet thickness is specified in the χMCF file.

This is computed by $\eta_i = \frac{d_i}{t_j}\sin\alpha_j$, where the variable i is specifying the weld index and the variable j is defined by the sheet index of the welded sheet related to the weld (α_j in case of a corner weld equals 90° and therefore $\sin\alpha_j=1$).

In the χMCF file, the following parameters can be specified (see Table 96):

Table 96 — Parameters of simple corner weld per `<weld_position/>` (w.p.)

Parameter	χMCF-Key	Multiplicity per w.p.	Value space	Use	Default value
a	thickness	1	≥ 0	Optional	
β	angle	1	≥ 0	Optional	45 [deg]
η	penetration	1	$0 \leq \eta \leq 1$	Optional	0

All other parameters are provided by the CAD or CAE model itself.

10.2.6.3 Double corner weld

10.2.6.3.1 Sheet parameters

The parameters to describe the connection are (see Figure 58):

— t_B Thickness of base sheet,

— t_1, t_2 Thicknesses of welded sheet,

— c_1, c_2 Gaps between base sheet and welded sheet,

— v_1, v_2 Misalignment of welded sheet.

10.2.6.3.2 Weld parameters

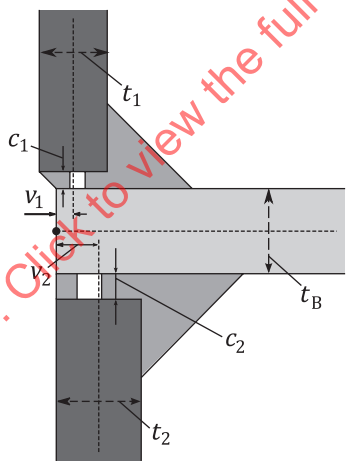


Figure 58 — Double corner weld sheet layout

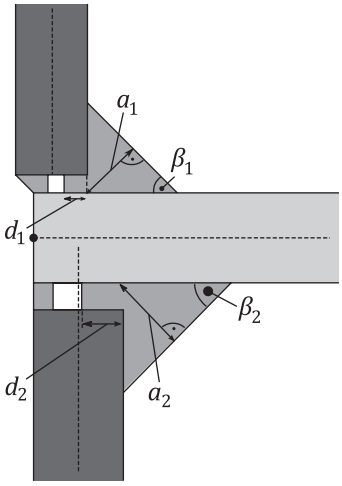


Figure 59 — Double corner weld parameters

The parameters of the welds are the same for all the potential physical welds on the connection (applies to Figure 59 above):

— a_i Thickness of the weld (a-value, throat),

— d_i Depth of the penetration,

— β_i Weld angle.

For the penetration, the ratio η_i of the penetration depth to the sheet thickness is specified in the χMCF file.

This is computed by $\eta_i = \frac{d_i}{t_j} \sin\alpha_j$, where variable *i* is specifying the weld index and variable *j* is defined by the sheet index of the welded sheet related to the weld (α_j in case of a corner weld equals 90° and therefore $\sin\alpha_j$=1).

In the χMCF file, the following parameters can be specified (see Table 97):

Table 97 — Parameters of double corner weld per `<weld_position/>` (w.p.)

Parameter	χMCF-Key	Multiplicity per w.p.	Value space	Use	Default value
a	thickness	1	≥ 0	Optional	
β	angle	1	≥ 0	Optional	45 [deg]
η	penetration	1	0 ≤ η ≤ 1	Optional	0

All other parameters are provided by the CAD or CAE model itself.

10.2.6.4 Attributes

10.2.6.4.1 Attribute `base`

The index for the base sheet is specified using the attribute `base`.

10.2.6.4.2 Attribute `technology`

The value for the attribute `technology` can be specified using the following values:

— `resistance`,

— arc,

— laser (energy beam / laser),

— friction,

— brazing.

10.2.6.5 Element <weld_position/>

For the element <weld_position/> the following attributes can be specified for the corner weld (see Table 98):

Table 98 — Attributes of element <weld_position/> for corner weld

Attributes	Type	Use
u	Floating point	Required
x	Floating point	Required
y	Floating point	Required
z	Floating point	Required
reference	Boolean	Optional
section	Selection	Optional
thickness	Floating point	Optional
angle	Floating point	Optional
shape	Selection	Optional
penetration	Floating point	Optional
filler	Selection	Optional
filler_material	Alphanumeric	Optional

10.2.6.5.1 Attributes u, x, y, z, and reference

The detailed definition is provided in 10.2.4.4.

10.2.6.5.2 Attribute section

Valid values for the attribute section of a corner weld are:

— HV,

— U,

— Fillet.

10.2.6.5.3 Attribute thickness

The attribute thickness specifies the thickness (a-value, throat) of the weld. Depending on the section this is required, optional or not allowed (see Table 99):

Table 99 — Values of attribute section

Attribute value “section"	Attribute “thickness"
HV	Optional
U	Not allowed
Fillet	Required

10.2.6.5.4 Attribute `angle`

The attribute `angle` specifies the angle of the weld relative to the base sheet. Depending on the section this is optional or not allowed (see Table 100):

Table 100 — Values of attribute `angle`

Attribute value “section"	Attribute “angle"
HV	Optional
U	Not allowed
Fillet	Required

10.2.6.5.5 Attribute `shape`

The attribute `shape` defines the shape of the weld throat. For the allowed values, see 10.2.4.4.21.

10.2.6.5.6 Attribute `penetration`

The attribute `penetration` specifies the degree of penetration resulting from the welding.

10.2.6.5.7 Attribute `filler`

Valid values for the attribute filler can be:

— `yes`,

— `no`.

Depending on the technology, the default value can differ, see 10.2.4.4.19 Attribute `filler`.

10.2.6.5.8 Attribute `filler_material`

The attribute `filler_material` specifies the applied material during the welding process.

EXAMPLE Definition of a `<corner_weld/>` with all attributes for the `<weld_position/>`:

```
<seamweld>
    <corner_weld base="1" technology="resistance">
        <weld_position u="0" x="0" y="1" z="0"
                       reference="false"
                       section="Fillet"
                       thickness="1.5"
                       angle="30"
                       shape="concave"
                       penetration="0.5"
                       filler="yes"
                       filler_material="E7018-X"/>
        <sheet_parameter ... />
    </corner_weld>
</seamweld>
```

10.2.6.6 Element `<sheet_parameter/>`

For the element `<sheet_parameter/>`, the following attributes can be specified for the corner weld (see Table 101):

Table 101 — Attributes of element `<sheet_parameter/>` for corner weld

Attributes	Type	Use	Constraint / Remarks
Index	Integer	Required	It shall be referenced to `<part/>` index attribute
Gap	Floating point	Optional	Default value is 0
sheet_thickness	Floating point	Optional	-
sheet_angle	Floating point	Optional	-

EXAMPLE

```
<seamweld>
    <corner_weld base="1" technology="resistance">
        <weld_position u="0" x="0" y="1" z="0" ... />
        <sheet_parameter index="2" gap="0" sheet_thickness="1.5" sheet_angle="90" />
    </corner_weld>
</seamweld>
```

10.2.7 Edge weld

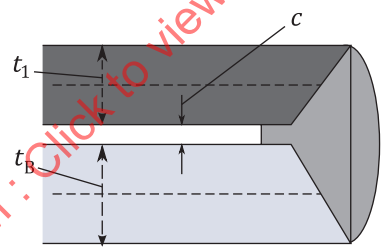
10.2.7.1 General

The principles of the modelling of edge welds for χMCF are described in this subclause. An edge weld describes a connection between two sheets welded at their forehead side.

The XML definition of an edge weld supports one position. The weld position is specified using the element `<weld_position/>` with the corresponding attributes and nested elements inside the subtype definition.

10.2.7.2 Sheet parameters

The parameters to describe the connection are (see Figure 60):



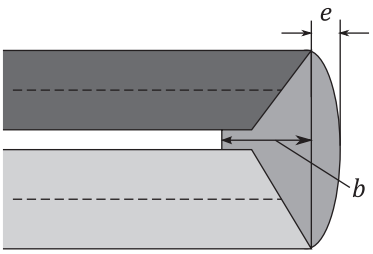
Key

t_B thickness of base sheet
t_1 thickness of welded sheet
c gap between base sheet and welded sheet

Figure 60 — Edge weld sheet layout

10.2.7.3 Weld parameters

The parameters of the weld are (see Figure 61):



Key

b width of the weld

e reinforcement

Figure 61 — Edge weld parameters

The following parameters can be specified for the edge weld (see Table 102):

Table 102 — Parameters of edge weld for its single `<weld_position/>`

Parameter	χMCF-Key	Multiplicity	Value space	Use	Default value
b	width	1	≥ 0	Optional	-
c	gap	1	≥ 0	Optional	0
e	-	1	≥ 0	Optional	0

NOTE The reinforcement is not yet supported as an attribute in χMCF version 3.1.1 and earlier.

10.2.7.4 Attributes

10.2.7.4.1 Attribute `base`

The index for the base sheet is specified using the attribute `base`.

10.2.7.4.2 Attribute `technology`

The value for the attribute `technology` can be specified using the following values:

— `resistance`,

— `arc`,

— `laser` (energy beam / laser),

— `friction`,

— `brazing`.

10.2.7.5 Element `<weld_position/>`

For the element `<weld_position/>` the following attributes can be specified (see Table 103):

Table 103 — Attributes of element `<weld_position/>` for Edge Weld

Attributes	Type	Use
u	Floating point	Required
x	Floating point	Required
y	Floating point	Required
z	Floating point	Required
reference	Boolean	Optional
section	Selection	Optional
width	Floating point	Optional
filler	Selection	Optional
filler_material	Alphanumeric	Optional

10.2.7.5.1 Attributes `u, x, y, z,` and `reference`

The detailed definition is provided in 10.2.4.4.

10.2.7.5.2 Attribute `section`

Valid values for the attribute `section` of an edge weld are:

— `I` (not be confused with seam weld subtype "`i_weld`", see10.2.4.1),

— `V`,

— `U`.

10.2.7.5.3 Attribute `width`

The attribute width specifies the `width` of the weld.

10.2.7.5.4 Attribute `filler`

Valid values for the attribute filler can be:

— `yes`,

— `no`.

Depending on the technology, the default value can differ, see 10.2.4.4.19 Attribute `filler`.

10.2.7.5.5 Attribute `filler_material`

The attribute `filler_material` specifies the applied material during the welding process.

EXAMPLE

```
<seamweld>
    <edge_weld base="1" technology="arc">
        <weld_position u="1" x="1" y="1" z="0"
            reference="false"
            section="V"
            width="2"
            filler="yes"
            filler_material="E7018-X"/>
        <sheet_parameter ... />
    </edge_weld>
</seamweld>
```

10.2.7.6 Element <sheet_parameter/>

For the element <sheet_parameter/>, the following attributes can be specified for the edge weld (see Table 104):

Table 104 — Attributes of element <sheet_parameter/> for edge weld

Attributes	Type	Use	Constraint / Remarks
Index	Integer	Required	It shall be referenced to <part/> index attribute
Gap	Floating point	Optional	Default value is 0
sheet_thickness	Floating point	Optional	-
sheet_angle	Floating point	Optional	-

EXAMPLE

```
<seamweld>
    <edge_weld base="1" technology="resistance">
        <weld_position u="1" x="1" y="1" z="0" ... />
        <sheet_parameter index="2" gap="0" sheet_thickness="1.5" sheet_angle="90" />
    </edge_weld>
</seamweld>
```

10.2.8 I-weld

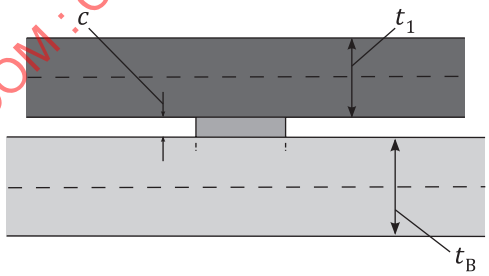
10.2.8.1 General

The principles of the modelling of I-welds for χMCF are described in this subclause. An I-weld describes a connection between two sheets welded together.

The XML definition of an I-weld supports one weld position. The weld position is specified using the element <weld_position/> with the corresponding attributes and nested elements inside the subtype definition.

10.2.8.2 Sheet parameters

The parameters to describe the connection are (see Figure 62):



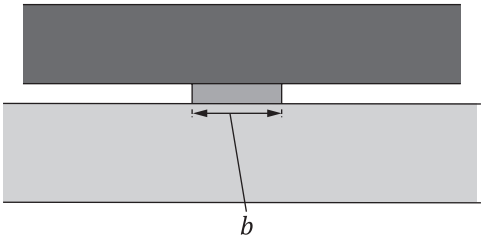
Key

t_B thickness of base sheet
t_1 thickness of welded sheet
c gap between base sheet and welded sheet

Figure 62 — I-weld sheet layout

10.2.8.3 Weld parameters

The parameters of the weld are (see Figure 63):



Key

b width of the weld

Figure 63 — I-weld parameters

The following parameter can be specified for the I-weld (see Table 105):

Table 105 — Parameters of I-weld for its single `<weld_position/>`

Parameter	χMCF-Key	Multiplicity	Value space	Use	Default value
b	width	1	≥ 0	Optional	-

All other parameters are provided by the CAD or CAE model itself and are partially used to specify parameters of the weld.

10.2.8.4 Attributes

10.2.8.4.1 Attribute `base`

The index for the base sheet is specified using the attribute `base`.

10.2.8.4.2 Attribute `technology`

The value for the attribute `technology` can be specified using the following values:

— `resistance`,

— `arc`,

— `laser` (energy beam / laser),

— `friction`,

— `brazing`.

10.2.8.5 Element `<weld_position/>`

For the element `<weld_position/>` the following attributes can be specified for the I-weld (see Table 106):

Table 106 — Attributes of element `<weld_position/>` for I-weld

Attributes	Type	Use
u	Floating point	Required
x	Floating point	Required
y	Floating point	Required
z	Floating point	Required
reference	Boolean	Optional

Table 106 *(continued)*

Attributes	Type	Use
width	Floating point	Optional
filler	Selection	Optional
filler_material	Alphanumeric	Optional

10.2.8.5.1 Attributes `u, x, y, z,` and `reference`

The detailed definition is provided in 10.2.4.4.

10.2.8.5.2 Attribute `width`

The attribute `width` specifies the width of the weld.

10.2.8.5.3 Attribute `filler`

Valid values for the attribute `filler` can be:

— `yes`,

— `no`.

Depending on the technology, the default value can differ, see 10.2.4.4.19 Attribute `filler`.

10.2.8.5.4 Attribute `filler_material`

The attribute filler_material specifies the applied material during the welding process.

EXAMPLE

```
<seamweld>
    <i_weld base="1" technology="laser">
        <weld_position u="0" x="1" y="1" z="1"
                       reference="false"
                       width="1.0"
                       filler="no"
                       filler_material="E7018-X"/>
        <sheet_parameter ... />
    </i_weld>
</seamweld>
```

10.2.8.6 Element `<sheet_parameter/>`

For the element `<sheet_parameter/>`, the following attributes can be specified for the I-weld (see Table 107):

Table 107 — Attributes of element `<sheet_parameter/>` for I-weld

Attributes	Type	Use	Constraint / Remarks
Index	Integer	Required	It shall be referenced to `<part/>` index attribute
Gap	Floating point	Optional	Default value is 0
sheet_thickness	Floating point	Optional	-

EXAMPLE

```
<seamweld>
    <i_weld base="1" technology="laser">
        <weld_position u="0" x="1" y="1" z="1" ... "/>
        <sheet_parameter index="2" gap="0" sheet_thickness="1.5"/>
    </i_weld>
</seamweld>
```

10.2.9 Overlap weld

10.2.9.1 General

The principles of the modelling of overlap welds for χMCF are described in this subclause. An overlap weld describes a connection between two, three or four sheets welded together.

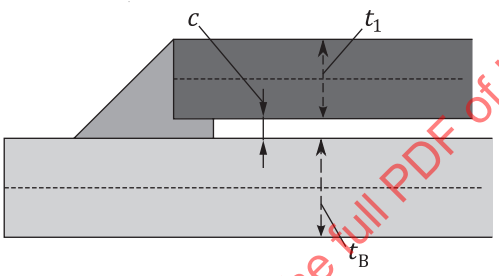
The XML definition of an overlap weld supports up to three weld positions. Each of the weld positions is specified using the element `<weld_position/>` with the corresponding attributes and nested elements inside the subtype definition.

NOTE Overlap welds with four sheets have been observed. However, they are not explicitly depicted in this document.

10.2.9.2 Simple overlap weld

10.2.9.2.1 Sheet parameters

The parameters to describe the connection are (see Figure 64):



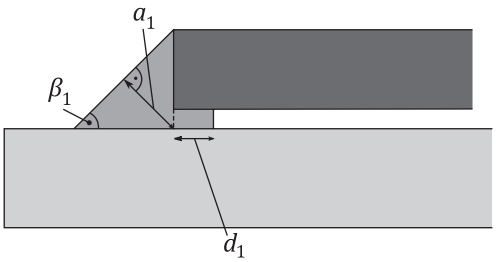
Key

t_B thickness of base sheet
t_1 thickness of welded sheet
c gap between base sheet and welded sheet

Figure 64 — Overlap weld sheet layout

10.2.9.2.2 Weld parameters

The parameters of the welds are the same for all the physical welds on the connection (see Figure 65):



Key

a_1 thickness of the weld (a-value, throat)
d_1 depth of the penetration
β_1 weld angle

Figure 65 — Overlap weld parameters

For the penetration, the ratio η_1 of the penetration depth to the sheet thickness is specified in the χMCF file.

This is computed by $\eta_1 = \frac{d_1}{t_1}$, where t_1 is the thickness of the attached sheet (green in Figure 65 above), not of the base sheet.

In the χMCF file, the following parameters can be specified (see Table 108):

Table 108 — Parameters of overlap weld per `<weld_position/>` (w.p.)

Parameter	χMCF-Key	Multiplicity per w.p.	Value space	Use	Default value
A	thickness	1	≥ 0	Optional	-
β	angle	1	≥ 0	Optional	45 [deg]
η	penetration	1	0 ≤ η ≤ 1	Optional	0

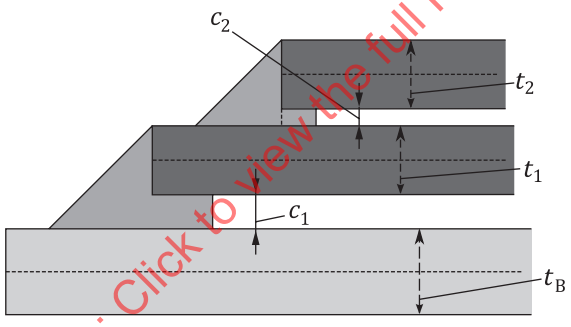
All other parameters are provided by the CAD or CAE model itself and are partially used to specify parameters of the weld.

10.2.9.3 Single sided double overlap weld

The single sided double overlap weld is represented by a stacked welding, see Figure 66.

10.2.9.3.1 Sheet parameters

The parameters to describe the connection are (see Figure 66):



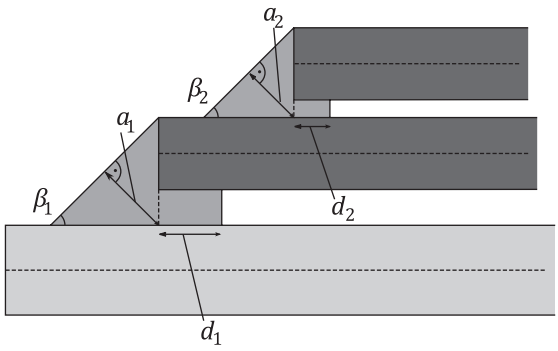
Key

t_B thickness of base sheet
t_i thicknesses of welded sheets
c_i gaps between sheets

Figure 66 — Single sided double overlap weld layout

10.2.9.3.2 Weld parameters

The parameters of the welds are the same for all the welds on the connection (see Figure 67):



Key

a_i thicknesses of the welds (a-value, throat)

d_i depths of the penetrations

β_i weld angles

Figure 67 — Overlap weld parameter details for lower (left) and upper (right) weld section

For the penetrations, the ratios η_i (i=1, 2) of the penetration depths to the sheet thicknesses are specified in the χMCF file.

They are computed by $\eta_i = \frac{d_i}{t_j} \sin\alpha_j$ where index i is specifying the weld index and index j is defined by the sheet index of the welded sheet related to the weld.

In the χMCF file, the following parameters can be specified (see Table 109):

Table 109 — Parameters of single-sided double overlap weld per `<weld_position/>` (w.p.)

Parameter	χMCF-Key	Multiplicity per w.p.	Value space	Use	Default value
A	thickness	1	≥ 0	Optional	-
β	angle	1	≥ 0	Optional	45 [deg]
η	penetration	1	0 ≤ η ≤ 1	Optional	0

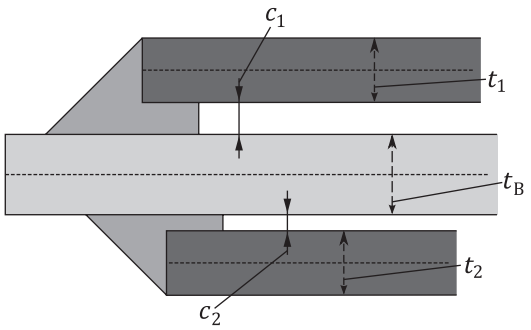
All other parameters are provided by the CAD or CAE model itself and are partially used to specify parameters of the weld.

10.2.9.4 Double-sided double overlap weld

A double-sided double overlap weld allows for welds on both sides of the base sheet, see Figure 68.

10.2.9.4.1 Sheet parameters

The parameters to describe the connection are (see Figure 68):



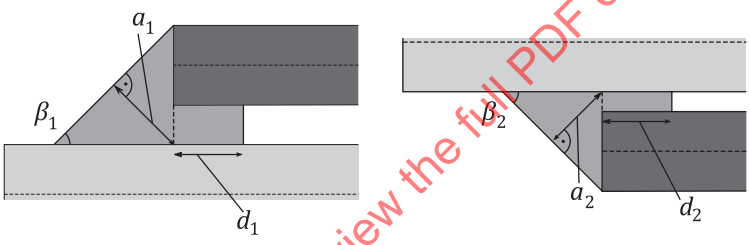
Key

t_B thickness of base sheet
t_i thicknesses of welded sheets
c_i gaps between base and welded sheets

Figure 68 — Double-sided double overlap weld layout

10.2.9.4.2 Weld parameters

The parameters of the welds are the same for all the welds on the connection (see Figure 69):



(left side: upper section — right side: lower section)

Key

a_i thicknesses of the welds (a-value, throat)
d_i depths of the penetrations
β_i weld angles

Figure 69 — Parameters of double-sided double overlap weld

For the penetrations, the ratios η_i (i=1, 2) of the penetration depths to the sheet thicknesses are specified in the χMCF file.

They are computed by $\eta_i = \frac{d_i}{t_j}\sin\alpha_j$ where index i is specifying the weld index and index j is defined by the sheet index of the welded sheet related to the weld.

In the χMCF file, the following parameters can be specified (see Table 110):

Table 110 — Parameters of double-sided double overlap weld per `<weld_position/>` (w.p.)

Parameter	χMCF-Key	Multiplicity per w.p.	Value space	Use	Default value
A	thickness	1	≥ 0	Optional	-
B	angle	1	≥ 0	Optional	45 [deg]
η	penetration	1	0 ≤ η ≤ 1	Optional	0

All other parameters are provided by the CAD or CAE model itself and are partially used to specify parameters of the weld.

10.2.9.5 Attributes

10.2.9.5.1 Attribute `base`

The index for the base sheet is specified using the attribute `base`.

10.2.9.5.2 Attribute `technology`

The value for the attribute `technology` can be specified using the following values:

— `resistance`,

— `arc`,

— `laser` (energy beam / laser),

— `friction`,

— `brazing`.

10.2.9.6 Element `<weld_position/>`

For the element `<weld_position/>`, the following attributes can be specified for the overlap weld (see Table 111):

Table 111 — Attributes of element `<weld_position/>` for overlap weld

Attributes	Type	Use
base	Integer	Optional
u	Floating point	Required
x	Floating point	Required
y	Floating point	Required
z	Floating point	Required
reference	Boolean	Optional
section	Selection	Optional
thickness	Floating point	Optional
angle	Floating point	Optional
shape	Selection	Optional
penetration	Floating point	Optional
filler	Selection	Optional
filler_material	Alphanumeric	Optional

10.2.9.6.1 Attributes `u, x, y, z,` and `reference`

The detailed definition is provided in 10.2.4.4.

10.2.9.6.2 Attribute `base`

For this type of weld, the `base` sheet can be specified inside the element `<weld_position/>`. This is necessary in the case of a stacked welding with two welded sheets.

10.2.9.6.3 Attribute `section`

The only valid value for the attribute `section` of an Overlap Weld is:

— `Fillet`.

This value is the default if the section attribute is not specified.

10.2.9.6.4 Attribute `thickness`

The attribute `thickness` specifies the thickness (a-value, throat) of the weld.

10.2.9.6.5 Attribute `angle`

The attribute `angle` specifies the angle of the weld relative to the base sheet.

10.2.9.6.6 Attribute `shape`

The attribute `shape` defines the shape of the weld throat. For the allowed values, see 10.2.4.4.21.

10.2.9.6.7 Attribute `penetration`

The attribute `penetration` specifies the degree of penetration resulting from the welding.

10.2.9.6.8 Attribute `filler`

Valid values for the attribute filler can be:

— `yes`,

— `no`.

Depending on the technology, the default value can differ, see 10.2.4.4.19 Attribute `filler`.

10.2.9.6.9 Attribute `filler_material`

The attribute `filler_material` specifies the applied material during the welding process.

EXAMPLE Definition of `<weld_position/>` with all attributes except `base`:

```
<seamweld>
    <overlap_weld base="1" technology="resistance">
        <weld_position u="0" x="0" y="0" z="1"
                       reference="false"
                       section="Fillet"
                       thickness="1.5"
                       angle="30"
                       shape="concave"
                       penetration="0.5"
                       filler="yes"
                       filler_material="E7018-X"/>
        <sheet_parameter ... />
    </overlap_weld>
</seamweld>
```

10.2.9.7 Element <sheet_parameter/>

For the element <sheet_parameter/>, the following attributes can be specified for the overlap weld (see Table 112):

Table 112 — Attributes of element <sheet_parameter/> for Overlap Weld

Attributes	Type	Use	Constraint / Remarks
index	Integer	Required	It shall be referenced to <part/> index attribute
gap	Floating point	Optional	Default value is 0
sheet_thickness	Floating point	Optional	-
sheet_angle	Floating point	Optional	-

EXAMPLE Definition of <sheet_parameter/> including optional parameters

```
<seamweld>
    <overlap_weld base="1" technology="resistance">
        <weld_position u="0" x="0" y="0" z="1"/>
        <sheet_parameter index="2" gap="1.0" sheet_thickness="1.5" sheet_angle="0"/>
    </overlap_weld>
</seamweld>
```

10.2.10 Y-joint

10.2.10.1 General

The principles of the modelling of Y-joints for χMCF are described in this subclause. A Y-joint describes a connection between two or three sheets. The Y-joint defines a connection between a welded sheet and a base sheet. There are two potential physical welds that can be specified for this type of connection. The parameters for each of the welds can be described separately.

The XML definition of a Y-joint supports up to three weld positions. Each of the weld positions is specified using the element <weld_position/> with the corresponding attributes and nested elements inside the subtype definition.

NOTE The two most common welding positions are shown in Figure 70. The third welding position would be from underneath the base sheet, using a laser.

10.2.10.2 Sheet parameters

The parameters to describe the connection are (see Figure 70):

— t_B Thickness of base sheet,

— t_1 Thickness of welded sheet,

— α Sheet angle of welded sheet,

— c Gap between base and welded sheet.

10.2.10.3 Weld parameters

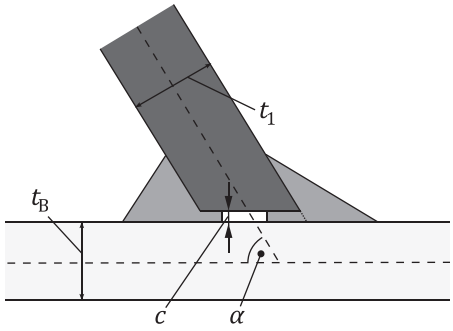


Figure 70 — Y-joint sheet layout

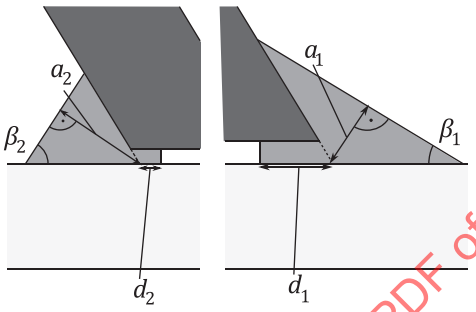


Figure 71 — Parameters of Y-joint

The parameters of the welds are the same for the four potential physical welds on the connection (applies to both subfigures of Figure 71):

— a_i Thicknesses of the welds (a-value, throat),

— d_i Depths of the penetrations,

— β_i Weld angles.

For the penetration, the ratio η_i of the penetration depth to the sheet thickness is specified in the χMCF file.

This is computed by $\eta_i = \frac{d_i}{t_j} \sin\alpha_j$, where index *i* is specifying the weld index and index *j* is defined by the sheet index of the welded sheet related to the weld.

In the χMCF file, only a subset can be specified (see Table 113):

Table 113 — Parameters of Y-joint per `<weld_position/>` (w.p.)

Parameter	χMCF-Key	Multiplicity per w.p.	Value space	Use	Default value
a	thickness	1	≥ 0	Optional	-
β	angle	1	≥ 0	Optional	45 [deg]
η	penetration	1	0 ≤ η ≤ 1	Optional	0

10.2.10.4 Attributes

10.2.10.4.1 Attribute `base`

The index for the base sheet is specified using the attribute `base`.

10.2.10.4.2 Attribute `technology`

The value for the attribute `technology` can be specified using the following values:

— `resistance`,

— `arc`,

— `laser` (energy beam / laser),

— `friction`,

— `brazing`.

10.2.10.5 Element `<weld_position/>`

For the element `<weld_position/>`, the following attributes can be specified for the Y-joint (see Table 114):

Table 114 — Attributes of element `<weld_position/>` for Y-joint

Attributes	Type	Use
base	Integer	Optional
u	Floating point	Required
x	Floating point	Required
y	Floating point	Required
z	Floating point	Required
reference	Boolean	Optional
section	Selection	Optional
thickness	Floating point	See attribute description
angle	Floating point	See attribute description
penetration	Floating point	See attribute description
filler	Selection	Optional
filler_material	Alphanumeric	Optional
shape	Selection	Optional

10.2.10.5.1 Attributes `u, x, y, z,` and `reference`

The detailed definition is provided in 10.2.4.4.

10.2.10.5.2 Attribute `base`

For this type of weld, the `base` sheet can be specified also inside the element `<weld_position/>`. This is necessary in the case of a stacked welding with two welded sheets.

10.2.10.5.3 Attribute `section`

The attribute `section` can be absent in the case of attribute value `technology="laser"` inside the element subtype.

Valid values for the attribute `section` (if present) of a Y-joint are:

— `Fillet`,

— `HV`,

— `HY`.

10.2.10.5.4 Attribute `thickness`

The attribute `thickness` specifies the thickness (a-value, throat) of the weld. Depending on the entry in `section` this is required, optional or not allowed (see Table 115):

Table 115 — Value dependency of attribute `thickness`

Attribute value "section"	Attribute "thickness"
HV	Optional
HY	Not allowed
Fillet	Required

10.2.10.5.5 Attribute `angle`

The attribute `angle` specifies the angle of the weld relative to the base sheet.

10.2.10.5.6 Attribute `penetration`

The attribute `penetration` specifies the degree of penetration resulting from the welding.

10.2.10.5.7 Attribute `shape`

The attribute `shape` defines the shape of the weld throat. For the allowed values, see 10.2.4.4.21.

10.2.10.5.8 Attribute `filler`

Valid values for the attribute `filler` can be:

— `yes`,

— `no`.

Depending on the technology, the default value can differ, see 10.2.4.4.19 Attribute `filler`.

10.2.10.5.9 Attribute `filler_material`

The attribute `filler_material` specifies the applied material during the welding process.

EXAMPLE Definition of a Y-joint with all parameters for two `<weld_positions/>`:

```
<seamweld>
    <y_joint base="1" technology="resistance">
        <weld_position u="0.5" x="1" y="0" z="1"
                       reference="false"
                       section="HY"
                       thickness="0.5"
                       angle="30"
                       penetration="0.5"
                       filler="yes"
                       filler_material="E7018-X"
                       shape="concave"/>
        <weld_position u="0.2" x="-1" y="0" z="1"
                       reference="false"
                       section="HY"
                       thickness="0.5"
                       angle="45"
                       penetration="0.5"
                       filler="yes"
                       filler_material="E7018-X"
                       shape="concave"/>
        <sheet_parameter ... />
    </y_joint>
</seamweld>
```

10.2.10.6 Element <sheet_parameter/>

For the element <sheet_parameter/>, the following attributes can be specified for the Y-joint (see Table 116):

Table 116 — Attributes of element <sheet_parameter/> for Y-joint

Attributes	Type	Use	Constraint / Remarks
index	Integer	Required	It shall be referenced to <part/> index attribute
gap	Floating point	Optional	Default value is 0
sheet_thickness	Floating point	Optional	-
sheet_angle	Floating point	Optional	-

EXAMPLE

```
<seamweld>
    <y_joint base="1" technology="resistance">
        <weld_position u="0.2" x="1" y="0" z="1" .../>
        <sheet_parameter index="2" gap="1.0" sheet_thickness="1.5" sheet_angle="180"/>
    </y_joint>
</seamweld>
```

10.2.11 K-joint

10.2.11.1 General

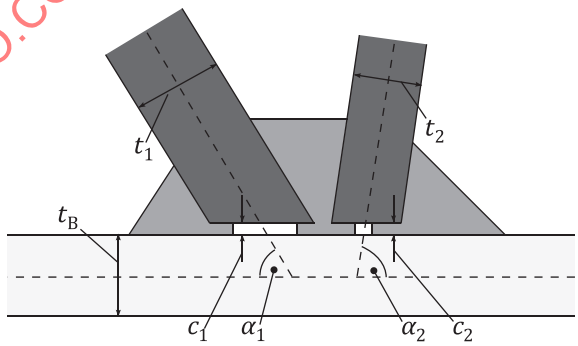
The K-joint connects two welded sheets from the same side to a base sheet.

There are four potential physical welds that can be specified for this type of connection. The parameters for each of the welds can be described separately. The three most common welding positions are shown in Figure 72. The fourth weld position would be from underneath the base sheet, using a laser.

The XML definition of a K-joint supports up to four weld positions. Each of the weld positions is specified using the element <weld_position/> with the corresponding attributes and nested elements inside the subtype definition.

10.2.11.2 Sheet parameters

The parameters to describe the connection are (see Figure 72):



Key

t_B thickness of base sheet
t_i thicknesses of welded sheets
α_i sheet angles of welded sheets
c_i gaps between base and welded sheets

Figure 72 — K-joint sheet layout