Varian

Medical Systems

TrueBeam™

Developer Mode

Version 2.0

User's Manual

(September 2013)

Developer Mode is intended for non-clinical use only and is NOT cleared for use on humans

This page intentionally blank

Reminder: Developer Mode is intended for non-clinical use only and is NOT cleared for use on humans

This page intentionally blank

Table of Contents

What is new in Version 2.0

TrueBeam version 2.0 includes a revised Developer Mode functionality, Developer Mode 2.0. The major feature introduced in Developer Mode 2.0 is a test suite intended for tracking i.e. the automatic machine motions needed to compensate for moving target volumes, for users with the Tracking License (TL) enabled.

In general, Developer Mode 2.0 TL allows for compensatory tracking motions using either real time couch movements or real time MLC shape offsets. See "Tracking Suite" section, near the end of this document, for more details.

The XML Beam syntax (schema) has thus been expanded to include tracking. Developer Mode 2.0 also brings some changes and many expansions to the gating parameter specifications/schema. These parameters were generally merged with the tracking parameters into a new Motion Management section. See also "Backwards compatibility" below for potential modifications necessary to run older Version 1.0, 1.5 and 1.6 Developer Mode XML beam scripts in Version 2.0.

Backwards Compatibility

Older XML scripts (written for Developer Mode 1.0, 1.5 and 1.6), especially those specifying gating parameters may have to be modified to become compatible with developer Mode 2.0. Following is a general list of changes to the XML Beam syntax/schema:

- kV Parameters:
  - The element <eFocalSpot> was renamed to <FocalSpot>
  - The element <eFluoroLevelControl> was renamed to <FluoroLevelControl>
  - The optional <FrameRate> element was added. If not defined, the highest frame is the default, which is currently 15?? fps.
  - The minimum mAs for single images was increased from 0.1mAs to 0.2mAs
- MovieParameters: Some elements were renamed to start with a capital letter. E.g. <width>was renamed to <Width>
- GatingParameters were expanded and moved to the new MotionManagementParameters section, which represents a more unified approach to motion management.
- The MotionManagementParameters section was added to encapsulate gating and tracking under motion management.
- Some additional attributes were defined for BeamHoldDevices, such as MVBeamImpact

General Warning

Developer Mode is intended for non-clinical use only and is NOT cleared for use on humans. In the descriptions below it became necessary to use some terms such as "patient", "treatment", "anatomy",

"implanted transponders" etc. to clarify certain research concepts in an effective manner. These terms refer to potential and eventual clinical use for future products that may be inspired, at least in part, from research performed using Developer Mode. These terms do not imply that developer Mode is intended for clinical use.

Introduction

This document describes how to use the Developer Mode research capabilities of the Varian TrueBeam linear accelerator. Developer Mode uses virtually the same control system as the TrueBeam Clinical modes. However, Developer Mode also enables access to additional more advanced control features which are not typically accessible or used by the Clinical Modes.

The user interface for Developer Mode is patterned after Service Mode. Developer Mode is permitted access to the full set of capabilities that have been built into the TrueBeam control system. Unlike the Clinical Modes, which are typically driven by plans downloaded through a DICOM interface, Developer Mode is driven by **XML Beams** loaded from local storage or network on the TrueBeam control console workstation computer. XML Beams are essentially text scripts in XML format where a rich instruction set allows Developer Mode users to construct and deliver complex non-standard beams, imaging and gating. In other words, it is this ability of Developer Mode to load and deliver XML beams that gives Developer Mode users full access to the complete dynamic beam, imaging and gating capabilities of the TrueBeam linear accelerator.

While the user interface (UI) for Developer Mode has a very similar look to Service Mode, the Developer Mode UI does not allow modifications to any machine configuration or operational parameters which may affect the functionality of Clinical Modes. So, for example, it is *not* possible to access the various accelerator subcomponents (*e.g.* functions such as beam and motor tuning) in Developer Mode. This general philosophy of Developer Mode guarantees that the normal operation of Clinical Modes cannot be affected by Developer Mode activities.

The following is a broad description of the additional control system features that are accessible through developer Mode:

- Define and deliver dynamic beams with arbitrary motion trajectories in MU-Position space for precisely controlled, predetermined variable dose rate, and simultaneous motion on potentially all accelerator axes (*e.g.* pre-programmed variable beam dose rate with simultaneous motion on gantry and/or potentially all collimation axes and/or potentially all couch axes).

- Program the acquisition of a variety of single images and/or start and stop several continuous imaging sequences at any predefined arbitrary point in the above beam/motion trajectory.

- Enable the respiratory gating signal to suspend/resume beam progression and/or imaging.

- A Tracking suite to experiment and research automatic couch and/or collimator (MLC) compensatory motions for moving anatomy (see "Tracking Suite" section).

As a reminder: Developer mode is intended strictly for non-clinical use. Developer Mode is NOT cleared for use on humans.

Support

Questions about True Beam Developer Mode can be sent to:

TrueBeamDeveloper@Varian.com

Please see MyVarian.com/Support/DeveloperMode for current examples and other helpful information.

Effect of Developer Mode use on Machine Performance

It is not possible to alter the clinical configuration of the TrueBeam machine in Developer Mode. There are a few configuration changes that can be made (mostly to some imaging parameters, such as MU per frame) but those changes are temporary, i.e., are reversed upon exiting Developer Mode.

However, any machine use may inevitably add to overall wear. In developer mode, it is possible to run long experiments with significant beam-on time as well as protracted aggregate moving parts motions. This may be particularly true for lengthy dynamic beam experiments, as well as tracking experiments where mechanical parts may be set in motion for extended periods of time. Remember take this into consideration as you design your experiments.

Operational Overview

In summary, the typical workflow in using Developer Mode consists of:

a) Using an editor to create XML documents conforming to the Varian TrueBeam XML Schema,
b) transferring the XML script files to the accelerator's control system
c) loading and running the XML script files through Developer Mode

The following figure shows the basic steps involved:

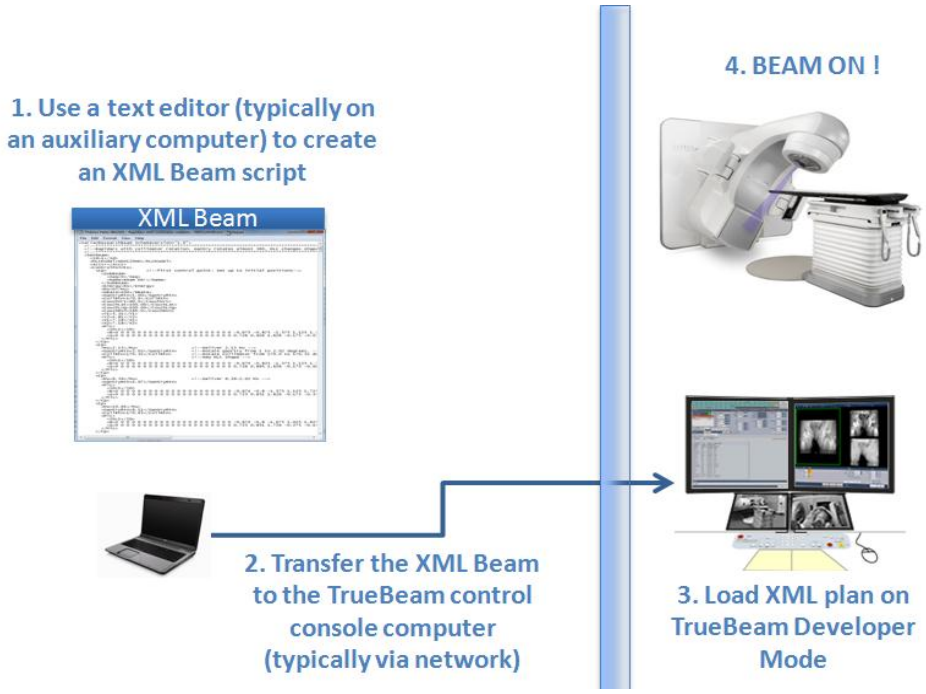


Fig 1. Elementary workflow of developing and delivering an XML beam

There are many methods of transferring the XML Beam file to the TrueBeam control console computer. However, transferring (copying) the file via network is recommended. Other methods, such as memory sticks, CDs and other digital media are more prone to infection with viruses and other malware. On TrueBeam accelerators, the I: drive is typically configured as a network drive and can be safely used to upload XML Beams to the accelerator.

In order to facilitate offline development, the typical workflow also includes two offline tools: an XML Beam Checker tool and an Emulator tool. These tools allow offline syntax checking and simulated delivery of XML Beams. These tools are described later in the document.

Access to Developer Mode

Access to Developer Mode is subject to license and user rights. Once a valid license has been issued, authorized users are allowed to launch Developer Mode by selecting “Research” from the TrueBeam control system main page and typing the user name and password.

Access rights for individual users are configured using the OSP Server platform as for Service Mode. It is recommended that users who need access to Developer Mode only (i.e. no acess to Service Mode) be given “Basic” user rights. The “Basic” level of access does not allow the overwriting of interlocks and faults which, in most cases, is adeguate for research purposes.

User Interface

As mentioned in the introduction, in terms of look and feel, the Developer Mode user interface is essentially a subset of the Service Mode user interface with one very powerful addition: the ability to load and deliver XML Beams.

Unlike Service Mode, those features that could affect machine performance in the Clinical Modes (e.g. Beam Tuning) have been disabled in Developer Mode. This is why, unlike Service Mode, Developer Mode does not provide low level access to the various accelerator sub-components. The purpose of this restriction is to keep Clinical Mode operations insulated from research performed using Developer Mode. The essential extra feature that Developer Mode provides compared to Service Mode is the ability to load and execute XML Beams.

Creating Varian XML Beams

Before we describe the details of how to create beam files using XML, it is important to first present the MU versus Position Trajectory Model used throughout the TrueBeam control system architecture.

The MU vs. Position Trajectory Model

The **MU versus Position Trajectory Model** or simply **Trajectory Model** is a central concept within the TrueBeam control system architecture and thus applies to all modes (Treatment, Service, Developer, QA etc.). According to the Trajectory Model, all beams, whether the typically more complex beams that are enabled only through Developer Mode or the simpler beams available in the Clinical and Service Modes, are all described as a relationship between MU and position [1].

In more mathematical terms, the Trajectory Model treats the position of all mechanical axes as a function of MU. So there is one function per axis which describes movements (if any) during beam delivery, *i.e.* (Axis Position) = F (MU). These functions are called **Trajectory Functions** and the entire ensemble of trajectory functions for all axes is abstractly referred to as **The Trajectory**. For those axes that never move throughout a beam, the Trajectory Function degenerates to a flat line.

[1] Note: The three imaging arms (MV imager, KV imager and KV source) can also be automatically moved during beams, however these arms do not follow the same exact dose vs. position model as the accelerator mechanical axes do. More details about moving the imaging arms can be found in the imaging section of this document.

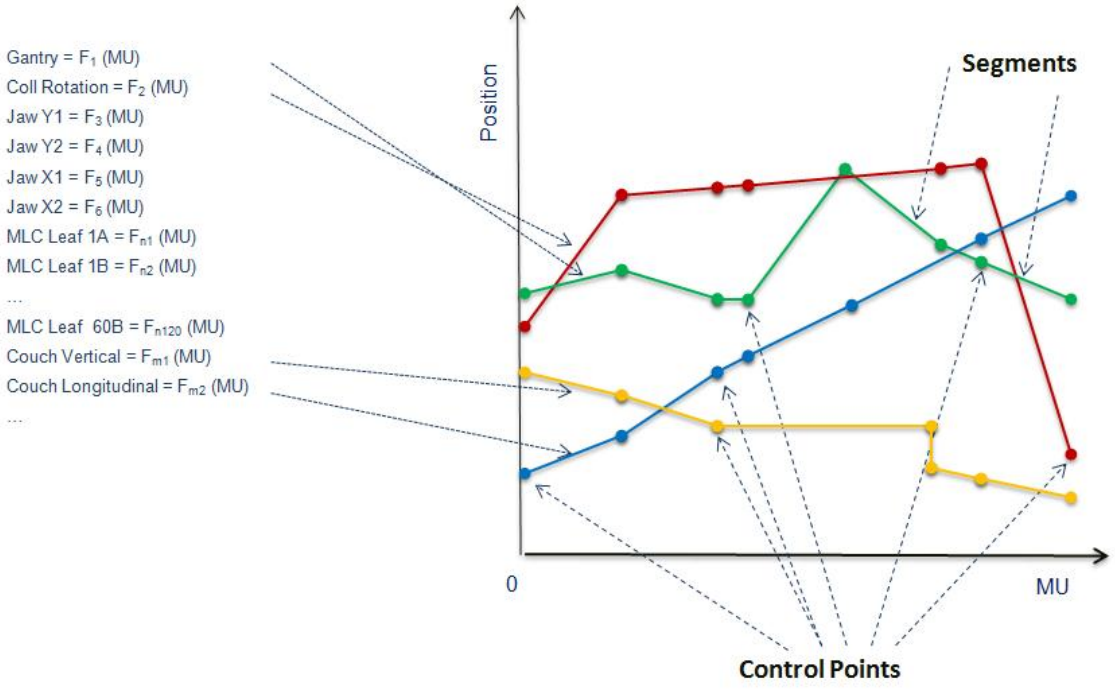


Fig 2. The MU vs. Position Trajectory model, Control Points and Segments

The Trajectory functions are segmented into linear segments. This facilitates representation of the function in digital form within the TrueBeam control system. In other words, the trajectory functions are described in terms of a finite number of discrete points called **Control Points**. The Developer Mode user (or any other planning entity for that matter) essentially programs the machine trajectory by specifying these Control Points in an XML Beam file.

The linear segment defined by two successive control points is simply called **Segment**. The piecewise-linear trajectory defined by the entire sequence of control points is referred to as the **Segmented Trajectory** or simply **Trajectory.** This piecewise-linear trajectory becomes the implied ideal trajectory to be followed at delivery time.

Another way to look at Control Points is that Control Points are essentially a mechanism to specify inflection points in the Trajectory *i.e.* points where the axis motion per MU delivered changes.

The sequencing convention for segments and control points is that the two successive control points *i-1* and *i*, define segment *i*. Therefore a beam starts with control point *0,* so control points *0* and *1* define segment *1*.

It is important to reiterate that there is an implied linear path within a segment and that the TrueBeam control system strictly enforces this implied linearity during beam delivery. The implied linearity is illustrated in the following example:

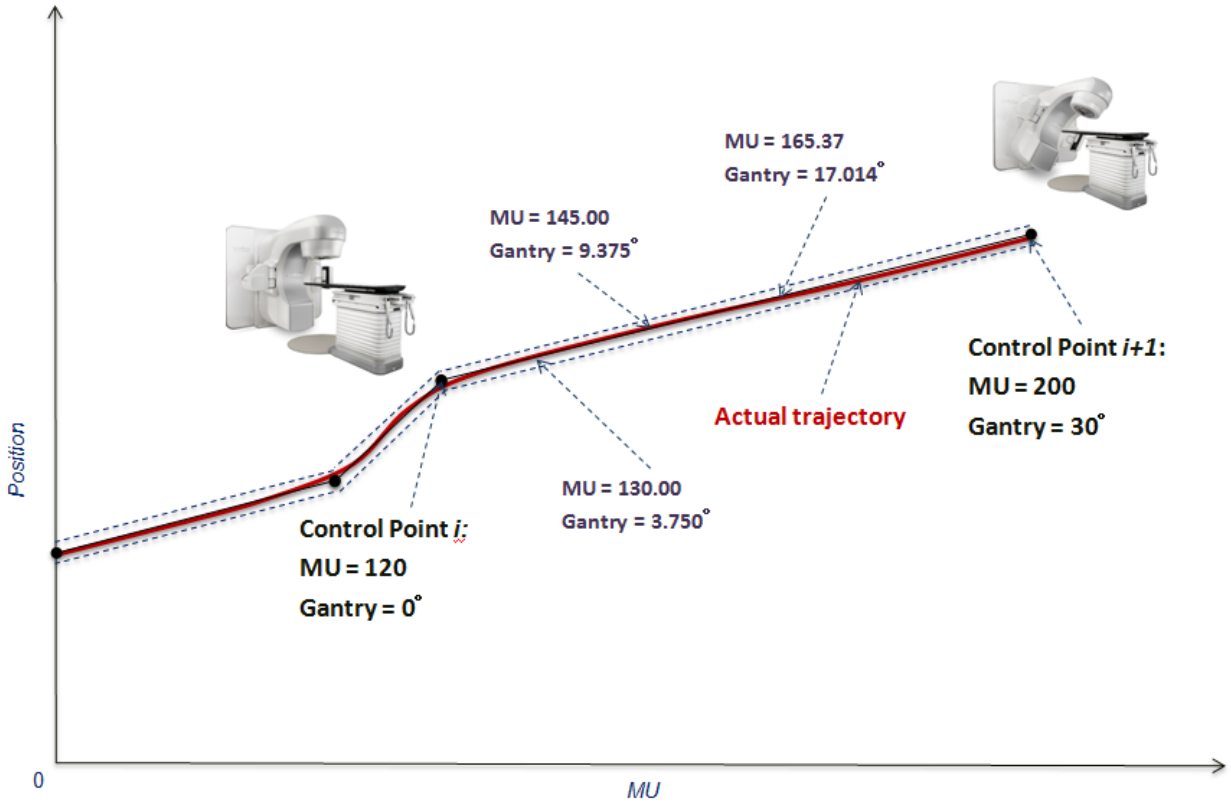


Fig 3. Trajectory and Linear Segment between Control Points.

The figure shows segment *i* defined by control points *i* and *i+1*.

By convention, beam delivery between control points is always specified in terms of cumulative MU. So, in the figure, segment *i* happens to define the arc like portion of a beam. During this arc segment, the gantry moves 30° (from position 0° to position 30°) while 80 MU (=200 MU- 120 MU) are delivered. The implied linearity means that the control system will control MU delivery and gantry angle so that the relationship between MU and gantry angle remains linear within any one segment. So, in the above example, when cumulative MU reaches 130 MU, the gantry will be at 3.750°, when the MU reaches 145 MU the gantry angle will be at 9.375° etc..

Notice that since the position of every mechanical axis is tied to MU, the relative position of axes with respect to each other is also precisely fixed during beam delivery. In other words, during beam delivery, motions follow an exact, predetermined, predictable and repeatable path, as specified in the XML Beam (the plan).

A segment that delivers radiation only without motion (a horizontal segment in the trajectory graph of figure 2) is referred to as a **Beam-Only Segment** or **Static Segment.**

A segment that simply moves one or more axes but delivers no MU (a vertical segment in the trajectory graph of figure 2, *i.e.* a segment defined by control points with the same cumulative MU) is referred to as a **Motion-Only Segment** or **Beamless Segment.** In principle, Motion-only segments are subject to the same standard linearity of motion principle, though there is a significant exception. The exception is that if a Motion only segment delivers neither MV nor KV radiation and there is MLC motion in the segment then the motions are not synchronized (see also "Motion-only segments may exhibit unsynchronized motion" in the "Limitations" section).

Composing XML beams

In principle, the only tool needed to use developer Mode is a text editor to compose XML files. There are also many commercially available XML editors that can simplify XML editing, but simple text editors are typically adequate.

The basic operation in TrueBeam Developer Mode is to compose XML files and then load them on the TrueBeam control system to deliver beam and/or take images. These XML instruction files are referred to as **XML Beam Files** or simply **XML Beams** and, as is typical of xml files, they have an .xml extension.

XML is a widely used standard to transfer information between computers. XML is text, so it is easily read and manipulated using a text editor. XML does not do anything by itself and imposes only a few simple syntax rules. XML is simply a way to transfer information packets, called "elements" in XML parlance. Typically these information packets/elements are organized into a hierarchical tree structure, however, no particular tree structure is imposed by XML itself. It is essentially up to the entities that exchange information using the XML standard to define and agree upon the particular tree structure(s) that will be used, and the meaning of the information contained in the elements.

More, easy to read, general information on XML is provided at http://www.w3schools.com/xml/. The site provides simple tutorials about XML (Note: This site is not controlled or endorsed by Varian in any way. Similar sites can also be found through a simple web search).

The Varian TrueBeam XML Schema

For the specific case of the TrueBeam machine, the specific XML information tree accepted by the TrueBeam control system is defined in the so called **Varian TrueBeam XML Schema** or simply **TrueBeam Schema**. The TrueBeam schema specifies the valid syntax and acceptable information content of XML Beam files. Studying the TrueBeam schema is a way to learn about all the capabilities of Developer Mode.

A Schema is another generic XML feature, which is a standard language used to describe and define the terms and structure that may be used (much as a dictionary and grammar is used for spoken languages). It is also written in XML but its elements contain specific keywords used to describe the structure of XML trees; such as elements and syntax details about which child elements are mandatory, which are

optional, the valid data types for an element etc.. A tutorial on XML schemas can be found at: http://www.w3schools.com/schema/default.asp .

Becoming familiar with the Varian XML Beam Schema is an essential part of doing research with Developer Mode. The Varian XML Beam Schema is described in the file **SetBeam.xsd**. The .xsd extension is typical of XML schema files. The **SetBeam.xsd** file is included in Appendix A. SetBeam.xsd can be used as a reference to the XML language accepted by TrueBeam. Varian also provides an electronic copy of SetBeam.xsd.

Upon examination of the Varian XML Beam schema, one sees that many of the elements in an XML Beam are optional, so the tree structure of XML beams can vary significantly from beam to beam. Simple fixed beams can be described as simple trees of just a few elements, while, in general, dynamic beams with more control points require more elements. Also, beams that take images are typically described by trees of greater depth since, in general, the information required to take images is more hierarchically organized.

The simplest example: Beam On

The following example is one of the simplest beams that can be written for the TrueBeam machine. When executed through developer mode, this beam simply delivers 100 monitor units of radiation.

```
<VarianResearchBeam SchemaVersion="1.0">
  <!--********************************-->
  <!--Just beam on for 100 MU          -->
  <!--********************************-->
  <SetBeam>
    <Id>1234</Id>
    <MLCModel>NDS120HD</MLCModel>
    <Accs></Accs>
    <ControlPoints>
      <Cp>
        <SubBeam>
          <Seq>0</Seq>
          <Name>Beam ON</Name>
        </SubBeam>
        <Energy>6x</Energy>
        <Mu>0</Mu>
        <DRate>300</DRate>
      </Cp>
      <Cp>
        <Mu>100</Mu>
      </Cp>
    </ControlPoints>
  </SetBeam>
</VarianResearchBeam>
```

Example 1: Just Beam On: One of the simplest XML beams.

The first element <VarianResearchBeam SchemaVersion="1.0"> is mandatory and the attribute SchemaVersion must currently be set to 1.0 as shown here. The purpose of this attribute is to prevent incompatible files from being loaded on the TrueBeam machine (*e.g.* should a later version of the TrueBeam machine, with a different and incompatible XML syntax, become available sometime in the future).

The <SetBeam> element is the overall container element for the beam and is also required in every XML beam file.

The <Id> element is an integer identifier for the entire beam. Developer Mode does not impose any restriction on how this field is used. Essentially, Developer Mode users are free to identify their XML beams as they see fit.

The <MLCModel> is also required for every beam. It lists the MLC model intended to be used with this beam. For example, the standard Milleneum 120 MLC is referred to as “NDS120” whereas the smaller-leaf HD MLC is referred to as “NDS120HD”. If the specified MLC model does not match the MLC model on the target machine the control system will prevent beam-on through an interlock.

The <Accs> element lists any accessories that must be mounted to deliver this beam. A list of the accessories accepted by TrueBeam and their corresponding codes is listed in Appendix B. The control system will prevent beam, through an accessory interlock, until the listed accessories are mounted. Leaving this element empty (as in this example) means that any accessory (including no accessories) is compatible with this beam (*i.e.* accessories are essentially “don’t care” in this case). The element must always be present even if left empty.

<ControlPoints> is the overall container for the control points that will define the MU vs. Position Trajectory for this beam.

<Cp> is the container for parameters associated with a control point. Since control points are used to specify the segments of the Trajectory function, a beam must contain at least 2 control points (*i.e.* one segment minimum). The first control point is special as it must contain some additional parameters (see below).

The <SubBeam> element is mandatory at the first control point. However the SubBeam element is typically of little interest to Developer Mode users.[2] It contains an integer sequence identifier and a generic string name.

<Energy> is a mandatory element in the first control point. It specifies the MV beam energy to be used. The format is “dds" where:

dd = 0-99 and

[2] The <SubBeam> is a construct used primarily by the clinical modes when radiation fields are combined into one automated field. However, since Developer Mode and the Clinical Modes use the same control system, this element is also present in Developer Mode.

s = 'x', 'e', or 'h', where,

x : MV X-Rays, e : MeV electrons, h : MeV HDTSe- electrons and

k : kV beams (used for beams that take kV images only and thus deliver no MV beam).

Examples: "6x" (for 6 MV X-Rays) "12e" (for 12 MeV electrons) and "0k" (for kV beam only).

The <Energy> element can appear only in the first control point. It is not possible to switch energies in the middle of an XML Beam. In other words, all MV radiation specified in a particular XML beam, is delivered at the same energy.

<Mu> is the cumulative MU for this control point. Since this is the first control point, the MU here must be zero.

<DRate> is the maximum dose rate to be used for this beam in Monitor Units per minute (300MU/min in this example). This element is mandatory in the first control point. For static beams the control system will deliver the Beam at this dose rate. However, for dynamic beams this dose rate is to be interpreted simply as a user defined “dose rate ceiling”, that is, a maximum dose rate not to be exceeded.

For dynamic beams where MU must be delivered concurrently with motions, the control system will generally compute and use a different dose rate on every segment. The segment dose rate used generally depends on which parameter is the limiting factor for that segment (*i.e.* MU or one of the concurrent motions). Therefore, in general, there is a maximum dose rate that can be used in a dynamic segment, depending on what else is moving in that same segment. If this maximum possible dose rate exceeds the ceiling dose rate specified by the user in the XML beam, then the control system will use the lower user-specified ceiling dose rate. Of course, the control system will then use also use proportionately lower speeds for the moving axes in the segment, in order to adhere to the MU vs. position Trajectory specified in the XML Beam.

It is important to reiterate that while the control system will never exceed the dose rate ceiling specified in the XML file, regardless of which one turns out to be the limiting factor on each segment (MU or one of the motions), the control system will always use an appropriate combination of dose rate and motion speeds so that the MU vs. Position Trajectory implied by the control points is satisfied. More information on his topic can be found in the section “The Time Element: Dose Rates, Axis Speeds and Beam Delivery Time” later in this document.

<Cp> is the container for the second control point. In this example, it contains only one element, <Mu>100</Mu> implying that 100MU will be delivered statically (*i.e.* radiation will be delivered while nothing moves).

In this particular example, all the positions of the mechanical axes, *i.e.* gantry, collimator rotation, MLC shape, couch positions etc. are not specified and are thus “don’t care”. This means that, for these axes, the control system will allow the user to beam on at any position.

Below are some additional, progressively more complicated XML beam examples. It is also possible to look at some of the XML beams that the TrueBeam control system itself generates. While Developer Mode cannot directly deliver DICOM plans, the Treatment modes convert every DICOM plan that is

loaded into the control system into an XML Beam. Therefore, any DICOM plan that is deliverable through the TrueBeam Treatment mode can also be translated in its equivalent XML Beam format. For more details see the section “Using a DICOM file as a starting point”.

Introducing static positions and tolerances

The following example adds enforcement of specific static positions and tolerances before the beam can be turned on.

```
<VarianResearchBeam SchemaVersion="1.0">
<!--*****************************************************************************-->
<!--200 MU at 400MU/min dose rate static field.                                  -->
<!--Requires that some axes are positioned within tolerance prior to beam on.    -->
<!--*****************************************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120HD</MLCModel>
    <TolTable>
      <GantryRtn>0.30</GantryRtn>
      <CouchVrt>2.00</CouchVrt>
      <CouchLng>1000.00</CouchLng>
      <CouchRtn>3.50</CouchRtn>
    </TolTable>
    <Accs></Accs>
    <ControlPoints>
      <Cp>
        <SubBeam>
          <Seq>0</Seq>
          <Name>Beam ON</Name>
        </SubBeam>
        <Energy>6x</Energy>
        <Mu>0</Mu>
        <DRate>400</DRate>
        <GantryRtn>221.00</GantryRtn>
        <CollRtn>180.0</CollRtn>
        <CouchVrt>90.54</CouchVrt>
        <CouchLng>100.00</CouchLng>
        <CouchRtn>180.0</CouchRtn>
        <Y1>5.31</Y1>
        <Y2>4.81</Y2>
        <X1>7.18</X1>
        <X2>7.18</X2>
        <Mlc>
          <ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.675 0.875 1.575 1.125 1.725
```

```
            2.225 2.725 3.225 3.425 1.025 3.325 2.925 0.425 1.825 1.425 0.075 0.275
            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.726 0.926 1.626 0.175 0.475
            0.475 0.075 0.525 0.625 0.325 1.025 1.825 2.025 1.625 1.025 0.525 0.326
            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
        </Mlc>
      </Cp>
      <Cp>
        <Mu>200</Mu>
      </Cp>
    </ControlPoints>
  </SetBeam>
</VarianResearchBeam>
```

Example 2: Adding static positions and tolerances.

This example delivers 200 MU statically. Nothing moves during beam delivery. However, this example enforces specific mechanical positions. By specifying positions for Gantry Rotation, Collimator Rotation, Couch Longitudinal, Couch Vertical, Couch Rotational, Collimator jaws Y1, Y2, X1, X2 and an MLC shape, this example will not allow beam on unless the mechanical axes listed are at their specified position.

Units: As a general rule, all positions in XML Beam files are always in Varian scale and the units are centimeters (cm) for translational axes and degrees for rotational axes. There is no limit to the number of decimal points that can be specified, however there is a practical positioning accuracy limit associated with the finite resolution and accuracy of the machine's position sensors.

If the position of an axis is not listed at a control point, it is assumed that it will not move and thus remain at the position established at some previous control point. This avoids repetition of positions for axes that do not move, and thus reduces the size of XML files required to specify a beam. This rule, however, does not apply for individual MLC leaves. If any of the MLC leaves must move at some control point then all the leaves must be included, even those that do not move. In other words, the MLC shape is always listed as a unit and the MLC shape can be omitted from a control point only when the entire MLC shape remains the same, *i.e.* none of the leaves move between the control points.

Note: While it is redundant to re-list across control points the same position for an axis that does not move, doing so does not render the beam invalid.

Another related rule is that if an axis is going to move at some point in the beam, then an initial position must be established for this axis at the first control point. This, of course, is true regardless of whether MU is simultaneously delivered while the axis is moving during beam delivery, or whether the motion is part of a beamless segment. As a corollary, if an axis is not listed in the first control point, then it cannot appear in any subsequent control points.

You may notice that the XML Beam of this example does not include a position for Couch Lateral. It is not necessary to list positions for all the axes. Some can be left as "don't cares", in which case the control system will not enforce any particular positioning for those axes prior to beam-on.

The above example also specifies tolerances for some of the mechanical axes. The control system will prevent beam on unless the axes are positioned to their listed positions within their specified tolerance. Therefore, in this example, the gantry would have to be positioned to within 0.3 degrees of its listed position of 221.00 degrees. The couch vertical would have to be positioned to within ±2.00cm of its specified 90.54 cm position (*i.e.* between 88.54 and 92.54 cm). The couch rotation would have to be positioned to within ±3.5 degrees from 180 (*i.e.* between 175.5 and 183.5 degrees).

Notice that collimator rotation has a position of 180.00 degrees listed but no tolerance has been specified for that axis. When no tolerance is specified for an axis, the default is that a tight internal control system tolerance applies by default. This tolerance is 0.1 cm (centimeters) for translational axes and 0.1 degrees for rotational axes. So, in this example, collimator rotation would have to be positioned at 180.00 ± 0.1 degrees for the system to allow beam on.

The Multileaf Collimator (MLC) must always be at the exact specified shape in order to beam on. In other words, the MLC always defaults to an internal tight tolerance. It is not possible to specify wider tolerances for the leaf positions.

In the above example, notice how the couch longitudinal tolerance has been set very wide. In essence that means that the control system will allow beam at any couch longitudinal position. The effect is similar to leaving the couch position unspecified as a don't care. However specifying a position, albeit with a large tolerance, provides a target position to easily drive to at delivery time (see automatic positioning of axes prior to beam on later in the document).

Notice that tolerances apply only to the initial position of static axes, that is, axes that do not move during beam delivery. For dynamic axes any specified tolerance becomes meaningless and the tight internal control system tolerance is applied for any axis which will move during beam delivery. This is true regardless of whether the axes will move concurrent with the beam or will move within the context of a motion only segment.

Finally, the position of static axes is actively frozen during beam delivery. The position is then monitored to within the internal tight tolerance of 0.1 cm or 0.1°. In the unlikely event that a static axis should move during beam delivery, a MOTN (Motion) interlock will occur.

Note on MLC leaf format: The positions of the leaves in the XML Beam must be separated by exactly one blank space. While the Beam Checker Tool allows a variable number of spaces, some TrueBeam control system versions require exactly one single space.

Note on MLC carriage positions: MLC leaves are mounted on a movable carriage. If the MLC will move while delivering the XML Beam, the control system automatically computes the optimal position of the carriages based on the MLC shape, or sequence of shapes. The carriages are then automatically moved into position together with the leaves when the user loads the XML Beam and performs a "Go To". If the sequence of MLC shapes requested is such that multiple carriage positions are required, then multiple carriage positions are automatically computed when the beam is loaded into the control system. Then, during Beam delivery, when the point is reached where a new carriage position is needed, then beam,

as well as all other motions, are temporarily suspended while the MLC carriages are moved to their new positions.

Adding static fields

The following example shows a Beam that delivers two static fields. The Beam-On button is pressed only once. After the MU for the first field is delivered (static segment), the motion to the next field happens automatically, without beam (motion only segment). Then upon reaching the second field, the beam comes on automatically.

While in this example the motion and the beam are not simultaneous, from a control system standpoint this is considered a dynamic beam, since, in a general sense, it requires the coordination of both beam and motion after the Beam-On button is pressed.

```
<VarianResearchBeam SchemaVersion="1.0">
  <!--*************************************************************************-->
  <!--Two static MLC fields delivered at two different gantry angles. First field
  delivers 110 MU, second field 72 MU    -->
  <!--*************************************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120HD</MLCModel>
    <Accs></Accs>
    <ControlPoints>
      <Cp>              <!--first control point: Set up to initial positions-->
        <SubBeam> <Seq>0</Seq> <Name>Beam ON</Name> </SubBeam>
        <Energy>6x</Energy>
        <Mu>0</Mu>
        <DRate>400</DRate>
        <GantryRtn>180.00</GantryRtn> <CollRtn>180.0</CollRtn>
        <CouchVrt>90.0</CouchVrt> <CouchLat>100.00</CouchLat>
        <CouchLng>100.00</CouchLng> <CouchRtn>180.0</CouchRtn>
        <Y1>5.31</Y1> <Y2>4.81</Y2> <X1>7.18</X1> <X2>7.18</X2>
        <Mlc>
          <ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.675 0.875 1.575 1.125 1.725
             2.225 2.725 3.225 3.425 1.025 3.325 2.925 0.425 1.825 1.425 0.075 0.275
             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.726 0.926 1.626 0.175 0.475
             0.475 0.075 0.525 0.625 0.325 1.025 1.825 2.025 1.625 1.025 0.525 0.326
             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
        </Mlc>
      </Cp>

      <!--second control point: Just deliver 110 MU without moving-->
```

```
      <Cp> <Mu>110</Mu></Cp>

      <!--third control point: No MU, just move gantry to new position and Mlc to
       new shape-->
      <Cp>
        <GantryRtn>90.00</GantryRtn>
        <Mlc>
          <ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.988 0.235 0.775 0.995 1.475
             1.525 1.725 2.825 2.525 3.625 3.825 1.325 3.725 3.125 0.725 1.925 1.620
            -0.170 -0.550 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -0.218 0.115 0.826 0.826 1.666
             0.215 0.775 0.975 0.275 0.525 0.825 0.725 1.025 1.725 2.325 2.025 1.425
             0.720 0.964 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
        </Mlc>
      </Cp>

      <!--fourth control point: Just deliver 72 MU (=182-110) without moving
      anything-->
      <Cp> <Mu>182</Mu> </Cp>
    </ControlPoints>
  </SetBeam>
</VarianResearchBeam>
```

Example 3: Adding static fields with MLC.

A Dynamic Beam

Below is a dynamic beam example. This beam is similar to a RapidArc®-type Beam except that it also rotates the collimator during irradiation. The gantry makes a full clockwise revolution from 360 to 0 degrees while MU is being delivered. As seen in the control points parameters, gantry rotation is not proportional to MU, therefore the MU is not uniformly distributed along the arc (*i.e.* variable MU per degree). However, within a segment (*i.e.* between any two successive control points) the behavior is linear, *i.e.* MU delivered, gantry motion, and change in MLC shape are all proportional. As previously mentioned, this intra-segment linearity is actively enforced by the control system.

```
<VarianResearchBeam SchemaVersion="1.0">
<!--*************************************************************************-->
<!--RapidArc with collimator rotation. Gantry rotates 360, MLC changes shape and also
collimator rotates.   -->
<!--*************************************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120</MLCModel>
```

```
    <Accs></Accs>
    <ControlPoints>
      <Cp>              <!--first control point: Set up to initial positions-->
        <SubBeam><Seq>0</Seq><Name>Beam ON</Name></SubBeam>
        <Energy>6x</Energy>
        <Mu>0</Mu>
        <DRate>400</DRate>
        <GantryRtn>360.00</GantryRtn><CollRtn>177.05</CollRtn>
        <CouchVrt>90.0</CouchVrt><CouchLat>100.00</CouchLat>
        <CouchLng>100.00</CouchLng><CouchRtn>180.0</CouchRtn>
        <Y1>5.31</Y1><Y2>4.81</Y2><X1>7.18</X1><X2>7.18</X2>
        <Mlc><ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -0.550 -0.875 -1.465 1.125
             1.835 2.125 2.555 3.625 3.775 1.225 4.325 2.925 1.325 1.725 1.525 -0.175
            -0.275 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.726 0.926 1.626 -0.175
            -0.575 -0.495 -0.175 0.725 0.825 0.465 1.025 1.825 2.025 1.625 1.025
             0.525 0.326 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
        </Mlc>
      </Cp>
      <Cp>
        <!--In this next control point, MU is being delivered uniformly as the
        Gantry and Collimator rotate, while MLC changes shape. -->
        <Mu>2.13</Mu>               <!--Deliver 2.13 MU -->
        <GantryRtn>357.03</GantryRtn> <!--Rotate gantry from 360 to 357.03 deg. -->
        <CollRtn>177.54</CollRtn> <!--Rotate collimator from 177.05 to 177.54 deg-->
        <Mlc>          <!--and linearly change to following MLC shape -->
          <ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -0.674 -0.825 -1.575 1.125
             1.725 2.225 2.670 3.200 3.425 1.025 3.325 2.925 0.425 1.825 1.425 -0.075
            -0.275 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.726 0.895 1.626 -0.175 –
             0.466 -0.475 -0.075 0.525 0.625 0.325 1.025 1.825 2.025 1.625 1.025
             0.525 0.326 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
        </Mlc>
      </Cp>
      <Cp>
        <Mu>6.09</Mu>                    <!--Deliver 6.09-2.13 MU -->
        <GantryRtn>355.07</GantryRtn>
        <Mlc>
          <ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -0.674 -0.8 -1.575 1.125 1.725
             2.225 2.625 3.125 3.325 1.425 3.225 2.925 0.818 1.825 0.925 -0.175 –
             0.174 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.725 0.851 1.626 -0.175 –
             0.375 -0.075 -0.075 0.525 0.725 -0.475 1.125 1.025 2.025 1.525 1.025
             0.525 0.225 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
```

```
        </Mlc>
      </Cp>
      …

      …Several more control points here…
      …
      …
      <Cp>
        <Mu>590.88</Mu>
        <GantryRtn>0.00</GantryRtn> <!--completed 360 gantry revolution.-->
        <Mlc><ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1.275 -1.475 -1.375 0.825 –
             2.975 -4.074 -4.274 3.125 3.425 3.525 3.425 3.125 -1.375 -0.482 1.425
             0.325 -0.775 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.326 1.526 1.426 1.918 3.125
             4.125 4.325 -2.675 -2.875 -1.975 -2.275 -2.275 2.225 1.525 1.025 0.325
             0.826 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
        </Mlc>
      </Cp>
    </ControlPoints>
  </SetBeam>
</VarianResearchBeam>
```

Example 4: A dynamic beam with simultaneous MU delivery and motions.

Note on imaging arms motions: An XML Beam can also specify motions for the three imaging arms (MV imager, kV Imager and kV Source). However, these mechanical axes cannot be moved while the MV beam is on and their motion does not follow the standard linear intra-segment trajectory pattern as the other mechanical axes do. This restriction is described in more detail in the section “Imaging Arm Positions, Tolerances and Motions”. When the imaging arms are moved, all other motions, as well as beam, are stopped while the arms move. Motions for the imaging arms are specified via the Imaging Parameters. Imaging parameters are described later in this document.

Note and limitation on the kV collimators (also known as kV blades), kV imaging foils and kV imaging filters: The current version of Developer Mode does not support motion of the kV collimators and/or changing of the kV imaging foils and/or kV imaging filters during beam delivery. Fixed positions for the kV collimators as well as a single set of kV imaging foils and kV imaging filters, to be used throughout the beam, can be requested in the XML Beam. However, these settings must be indirectly established prior to beam-on. For more details on this limitation see the section “Indirectly Setting Up the kV Foils, kV Filters and kV Blades” later in this document.

MLC Carriage Movements

Is there meant to be text here?

Specifying Dose Rate and Velocity limits

All the previous XML beam examples contain a Dose Rate specification. As already mentioned, in general, this dose rate should be interpreted as a limit, a ceiling. Therefore, the dose rate parameter is better referred to as a **Dose Rate Ceiling**. As the name implies this is a limit setting which enables the user to instruct the control system to never exceed the specified dose rate limit. As already explained, in general, it is not possible to specify an absolute fixed target dose rate for a dynamic beam. This is because, in general, for a dynamic beam, there are other instantaneous factors such as motions or image acquisitions that may indirectly impose a lower dose rate. See also the section "The Time Element: Dose Rates, Axis Speeds and Beam Delivery Time" later in this document.

Similarly it is possible to specify a **Velocity Table** inside an XML Beam. The Velocity Table contains velocity limits for the various mechanical axes. These, again, are simply velocity limit/ceiling settings which enable the user to instruct the control system to never exceed the specified velocities[3]. In general, the actual instantaneous velocity of an axis may be lower than the set limit. This is because the actual instantaneous velocity within a segment will generally depend on other motions taking place or beam being delivered in the same segment[4].

The following XML Beam portion shows how a gantry velocity limit of 2.5 degrees per second and a dose rate ceiling limit of 520 MU/min may be specified for the previous RapidArc® example:

```
<VarianResearchBeam SchemaVersion="1.0">
<!--************************************************************************-->
<!--RapidArc with collimator rotation. Gantry rotates 360, MLC changes shape and also
collimator rotates.   -->
<!--************************************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120</MLCModel>
    <VelTable>
      <GantryRtn> 2.50 </GantryRtn>
    </VelTable>
    <Accs></Accs>
    <ControlPoints>
      <Cp>              <!--first control point: Set up to initial positions-->
        <SubBeam><Seq>0</Seq><Name>Beam ON!</Name></SubBeam>
```

[3] From a conceptual standpoint, when dose is viewed simply as another axis, dose rate is similar to velocity and thus the dose rate limit/ceiling is similar to the velocity limits/ceilings. They are all simply limits on rate of change.

[4] Imaging may also have some effect on velocities, as beam progression may have to be throttled down or suspended during certain phases of the image acquisition process.

```
        <Energy>6x</Energy>
        <Mu>0</Mu>
        <DRate>520</DRate>
        <GantryRtn>360.00</GantryRtn><CollRtn>177.05</CollRtn>
        <CouchVrt>90.0</CouchVrt><CouchLat>100.00</CouchLat>
        <CouchLng>100.00</CouchLng><CouchRtn>180.0</CouchRtn>
        <Y1>5.31</Y1><Y2>4.81</Y2><X1>7.18</X1><X2>7.18</X2>
        <Mlc><ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -0.550 -0.875 -1.465 1.125
             1.835 2.125 2.555 3.625 3.775 1.225 4.325 2.925 1.325 1.725 1.525 -0.175
            -0.275 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.726 0.926 1.626 -0.175
            -0.575 -0.495 -0.175 0.725 0.825 0.465 1.025 1.825 2.025 1.625 1.025
             0.525 0.326 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
        </Mlc>
      </Cp>
      …
      …
      …
```

Example 5: Specifying Dose Rate and Velocity Limits

As mentioned, the limit will ensure that gantry speed will not exceed 2.5 degrees per second. However, the actual instantaneous gantry speed chosen by the control system may be lower for those segments of the RapidArc® which may involve delivery of large number of MU, large MLC leaf movements or, in this case, a lot of collimator rotation.

The velocity limit is global within the beam. The velocity limit for an axis will apply regardless of whether MU is delivered concurrently with motion or whether the motion happens within the context of a motion only segment (*i.e.* beamless segment). Specifying velocity limits for axes that never move during the beam has no effect.

The Time Element: Dose Rates, Axis Speeds and Beam Delivery Time

The XML beams do not contain a time element. In other words, it is not possible for the user to specify the exact time progression and overall time required to deliver an XML beam.

This is because it would be quite complicated for users to evaluate all the trajectory and dose interdependencies present in a dynamic beam. It is thus the machine, which has knowledge of its limitations (eg. dose rate, speeds, accelerations) that is best suited to convert the dose position relationship implied in a specific XML beam into a Dose-Position-Time trajectory.

As mentioned, users can exercise some limited control in the timing by specifying dose rate ceilings and velocity limits for various axes in their XML beams. However, there is no direct user control of the time

element. The XML beam remains to a large extent simply the specification of an MU vs. position trajectory. The time element is eventually computed by the control system when the XML beam is loaded for delivery.

In an abstract way, by introducing the time element, the control system takes the XML MU vs. position trajectory and converts it into an MU vs. time and position vs. time trajectory. This conversion takes place a-priori, that is, before beam on, when the XML beam is loaded into the control system (to be more exact when the "PREPARE" button is pressed on the TrueBeam machine's Control Console).

In general, the control system will compute the time progression so that the beam is delivered as fast as possible given

(a) the inherent machine limits (*e.g.* maximum possible dose rates, maximum possible speeds, maximum accelerations etc.)
(b) the user specified MU vs. position trajectory in the XML beam, and
(c) any additional user specified limits such as a dose rate ceiling and axis velocity limits.

The control system computes the dose rate and axis velocities on a per segment basis. For every segment, the control system analyzes the number of MU to be delivered and total axis movement for that segment. It then determines what is referred to as the **Limiting Rate** for that segment. The limiting rate is the rate that limits how fast a segment can be delivered. The limiting rate may result to be either the dose rate or the maximum velocity (motion rate) of an axis. These, in turn, may result to be either the inherent machine maximum dose rate and inherent machine axis velocities or the (lower) user specified Dose Rate Ceiling and/or velocity limits. The control system uses the maximum possible rate for the Limiting Rate and then computes proportionally the segment dose rate and/or segment velocity of the other axes necessary to maintain the user specified MU vs. position trajectory as specified in the XML Beam file, while still respecting any user imposed dose rate or velocity limits.

Generally, the limiting rate can vary from segment to segment. So, for example, it may turn out that a segment where the limiting rate is gantry velocity may be followed by a segment where the limiting rate is dose rate or the maximum velocity of another mechanical axis, eg. one of the MLC leaves.

The control system also contains algorithms to more precisely control the finite accelerations of various mechanical axes. Since the accelerator's mechanical parts have mass and associated inertia, it is generally not possible to instantaneously change their speed at segment transitions (*i.e.* at control points). The control system employees various more sophisticated algorithms to handle these segment transitions, so that the user specified MU vs. position trajectory as specified in the XML beams is followed at all times, including segment transitions.

The total (wall clock) time to deliver a beam is simply the sum of the times required to deliver every segment. However, note that the time counter shown on the Developer Mode user interface does not account for time spent in beamless segments (*i.e.* motion only segments). Therefore, when delivering an XML Beam, the time shown in the counter will not match the wall clock time for those beams that contain one or more motion only segments (*i.e.* the counter will typically show less than wall clock time in these cases).

Velocity Limits

The following maximum velocities apply to any XML Beam that moves one or more axes dynamically. Unless, the XML beam itself specifies a lower velocity limit (ceiling) then the following values are used to calculate the position vs. time trajectory for all moving axes. As explained in the previous section, the actual instantaneous velocity observed at any point in the beam may be lower than the limits below or any lower limit specified by the user in the XML Beam itself, because of other constraints such as dose delivered or other moving axes.

Axis	Max Velocity
MLC Carriage	1.20 cm/sec
MLC Leaf	2.50 cm/sec
GantryRtn	6.00 deg/sec
Collimator Rtn	15.00 deg/sec
CouchVrt	2.00 cm/sec
CouchLat	4.00 cm/sec
CouchLng	8.00 cm/sec
CouchRtn	3.00 deg/sec
Coll X1	2.40 cm/sec
Coll X2	2.40 cm/sec
Coll Y1	2.40 cm/sec
Coll Y2	2.40 cm/sec

Imaging

The following sections describe how to add imaging to the XML Beam files.

Kilo-Voltage and Mega-Voltage images can be taken anywhere in the XML beam. In the most general case, Kilo-Voltage (kV) and Mega-Voltage (MV) images are taken either before therapeutic MV MU is delivered and/or after therapeutic MV MU is delivered and/or interspersed into a beam while therapeutic MV MU is delivered or axes are moved in motion-only segments. It is also often desirable to construct XML Beams that take images only. We discuss the more general case first: XML Beams that take images concurrent with the MV therapeutic beam.

The Imaging Parameters Section

Imaging instructions inside an XML Beam are given via <ImagingPoints> which are logically equivalent to the <ControlPoints> that are used to specify the MU vs. position trajectory of the therapeutic MV beam. However, the <ImagingPoints> are specified after the <ControlPoints> in a separate section called

<ImagingParameters>. The imaging points are then synchronized to the therapeutic beam by referencing the <ControlPoints> from inside the <ImagingPoints> (see following examples).

Example: Acquiring two MV and two kV images concurrent with beam

The following example shows an XML Beam that delivers a 180 MU, 200 degree gantry rotation arc field. The Beam contains more than two control points to make the number MU per degree vary throughout the arc (*i.e.* inflection points are needed to describe non-linear behavior). Five regions of different MU per degree radiation density comprise the arc. Two MV images are taken concurrent with the beam, one at the 10 MU point and one at the 172.79 MU point. Also two kV images are taken, one at the 20 MU point and one at the 120 MU point.

While, generally speaking, the image acquisitions are, in a sense, concurrent with the MV beam, typically, image taking will interfere briefly with beam delivery. This is because the imager and beam delivery must "handshake"[5], as the imaging system must be notified that the imaging point has been reached, the MV beam must be suspended while reading images taken in certain modes etc. Typically, these handshake operations are short and only introduce minor delays into the XML beam delivery. In the following example, a very short pause (slowdown) is seen in both the gantry speed and the MV beam delivery while the 4 images are being acquired. However, these slowdowns are already planned by the control system a priori (*i.e.* before beam on). Therefore, in this example, the short gantry slowdown is synchronized with beam so that the implied linearity of MU vs. gantry angle (implied in the arc) is maintained at all times. In other words, in this example, MU vs. position linearity in the arc segment is maintained at all times regardless of any imaging related temporary slowdowns.

```
<VarianResearchBeam SchemaVersion="1.0">
<!--*************************************************************************-->
<!--Arc Beam, delivers 180 MU over 200 degree angle.
    5 arc regions at different MU/deg.
    MV images taken at 10 MU and 172.79 MU
    KV images taken at 20 MU and 120 MU
-->
<!--*************************************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120HD</MLCModel>
    <Accs></Accs>
    <ControlPoints>
      <Cp> <!--control point 0 (zero).-->
```

[5] Portions of the handshake can be eliminated through the <handshake> parameters in the imaging section. However, the handshake cannot be completely eliminated even if the <handshake> parameter is set to "false".

```
        <SubBeam><Seq>0</Seq><Name>Arc N Image</Name></SubBeam>
        <Energy>6x</Energy><Mu>0</Mu><DRate>300</DRate><GantryRtn>0.00</GantryRtn>
      </Cp>
      <!--Control points 1,2,3,4,5 follow...-->
      <Cp> <Mu>40</Mu> <GantryRtn>40.00</GantryRtn> </Cp> <!--40 deg @1    MU/deg.-->
      <Cp> <Mu>100</Mu><GantryRtn>80.00</GantryRtn> </Cp> <!--40 deg @1.5  MU/deg.-->
      <Cp> <Mu>140</Mu><GantryRtn>100.00</GantryRtn></Cp> <!--20 deg @2.0  MU/deg.-->
      <Cp> <Mu>170</Mu><GantryRtn>160.00</GantryRtn></Cp> <!--60 deg @0.5  MU/deg.-->
      <Cp> <Mu>180</Mu><GantryRtn>200.00</GantryRtn></Cp> <!--40 deg @0.25 MU/deg.-->
    </ControlPoints>

    <!--This is the imaging instructions section of the beam.-->
    <ImagingParameters>
      <DuringTreatment />
      <ImagingPoints>
        <!--*********************************-->
        <!--Instructions for first MV image. -->
        <!--*********************************-->
        <ImagingPoint>
          <Cp>0.25</Cp> <!--Index image, to be taken at 10 MU point.-->
          <Acquisition>
            <AcquisitionId>1</AcquisitionId>
            <AcquisitionSpecs /> <!--will use default specs-->
            <AcquisitionParameters>
              <ImageMode id="Highres" />
              <CalibrationSet>DefaultCalibrationSetId</CalibrationSet>
              <MV /> <!--this is an MV image.-->
            </AcquisitionParameters>
          </Acquisition>
        </ImagingPoint>

        <!--*********************************-->
        <!--Instructions for first KV image. -->
        <!--*********************************-->
        <ImagingPoint>
          <Cp>0.5</Cp><!--Index image, to be taken at 20 MU point.-->
          <Acquisition>
            <AcquisitionId>2</AcquisitionId>
            <AcquisitionSpecs>
              <Handshake>true</Handshake>
              <KV>true</KV>
            </AcquisitionSpecs>
            <AcquisitionParameters>
              <ImageMode id="HighQuality">
              </ImageMode>
              <CalibrationSet>DefaultCalibrationSetId</CalibrationSet>
              <KV>
```

```
            <KiloVolts>80</KiloVolts>
            <MilliAmperes>50</MilliAmperes>
            <MilliSeconds>10</MilliSeconds>
            <FluoroLevelControl>None</FluoroLevelControl>
          </KV>
        </AcquisitionParameters>
      </Acquisition>
    </ImagingPoint>

    <!--**********************************-->
    <!--Instructions for second KV image. -->
    <!--**********************************-->
    <ImagingPoint>
      <Cp>2.5</Cp><!--Index image, to be taken at 120 MU point.-->
      <Acquisition>
        <AcquisitionId>3</AcquisitionId>
        <AcquisitionSpecs>
          <Handshake>true</Handshake>
          <KV>true</KV>
        </AcquisitionSpecs>
        <AcquisitionParameters>
          <ImageMode id="HighQuality">
          </ImageMode>
          <CalibrationSet>DefaultCalibrationSetId</CalibrationSet>
          <KV>
            <KiloVolts>70</KiloVolts>
            <MilliAmperes>15</MilliAmperes>
            <MilliSeconds>10</MilliSeconds>
            <FluoroLevelControl>None</FluoroLevelControl>
          </KV>
        </AcquisitionParameters>
      </Acquisition>
    </ImagingPoint>

    <!--**********************************-->
    <!--Instructions for second MV image. -->
    <!--**********************************-->
    <ImagingPoint>
      <Cp>4.279</Cp> <!--Index image, to be taken at 172.79 MU point.-->
      <Acquisition>
        <AcquisitionId>4</AcquisitionId>
        <AcquisitionSpecs />
        <AcquisitionParameters>
          <ImageMode id="Highres" />
          <CalibrationSet>DefaultCalibrationSetId</CalibrationSet>
          <MV />
        </AcquisitionParameters>
```

```
          </Acquisition>
        </ImagingPoint>

      </ImagingPoints>
      <ImagingTolerances /> <!--mandatory, even if empty-->
    </ImagingParameters>

  </SetBeam>
</VarianResearchBeam>
```

Example 6: Acquiring kV and MV images concurrent with MV beam.

The <DuringTreatment /> element signifies that the images are part of an XML Beam that also delivers therapeutic MV radiation. **NOTE:** Some elements contain the word "treatment" because the same XML scripts are internally used by the control system to deliver beam in the clinical modes. However, as a reminder: **Developer Mode is intended for non-clinical use only and is NOT cleared for use on humans.**

Specifying image acquisition points:

Notice how, in general, the imaging points reference the control points in terms of fractional control points. So, the <Cp>0.25</Cp> element in the first MV image is what indexes the image acquisition to the MV beam. It essentially tells the control system to take the MV image 0.25 into the 1st segment, *i.e.* fraction 0.25 along the way from control point 0 to control point 1. This is the point where the MU reaches 10 MU. Similarly element <Cp>4.279</Cp> in the second MV image tells the control system to take the image at fraction 0.279 into the 5th segment (*i.e.* transition from control point 4 to control point 5). At fractional control point 4.279 the MU will be 172.79 MU[6].

The first MV image:

The <AcquisitionId>1</AcquisitionId> is a user specified unique integer identifier for the image. Typically it is desirable to assign a unique identifier to each image acquisition within an XML Beam.

<AcquisitionSpecs /> is an element that enables the user to change the default acquisition specifications. In this example the default specifications for the imaging mode specified will be used, therefore the element is left empty.

The <ImageMode id="Highres" /> element specifies the image mode. In this case High Resolution (also referred to as High Quality). There are other MV imaging modes available, such as "Continuous", "Dosimetry", "Triggered" and "Lowres" (also referred to as Low Dose). These modes correspond to those available in the Treatment and Service Modes. The Treatment imaging modes are also described in the "TrueBeam Technical Reference Guide Volume 2: Imaging" document. See also the section "Imaging Modes" below.

[6] Note that MU precision is typically limited to the quantization imposed by integral beam pulses. The MU per beam pulse varies by energy. As an example, for 6MV X-Rays, each pulse delivers approximately 0.03 MU. Therefore dose precision at 6X is quantized around 0.03 MU. For the FFF energy modes MU per pulse and the resulting quantization is coarser (up to ~0.11MU).

<CalibrationSet>DefaultCalibrationSetId</CalibrationSet> signifies that the default calibration set will be used for this image acquisition.

The first kV image:

The first kV image is taken half way into the first segment as implied by the <Cp>0.5</Cp> element. That computes to the point when the MU counter reaches 20 MU.

The <Handshake>true</Handshake> element allows the image exposure and image reading to be synchronized with MV beam delivery. While it is possible to take images where MV radiation and image acquisition (either kV or MV) are not synchronized, it is generally recommended that this parameter be set to true.

The <KV>true</KV> element signifies that this is a kV image.

<ImageMode id="HighQuality"> is the kV imaging mode. In this case “High Quality”. There are other kV imaging modes available, such as “DynamicGain”, “DynamicGainFluoro”, “DynamicGainLowFramerate”, “DynamicGainLowFramerateFluoro”, “LowDose”, “Triggered” and “Fluoro”. These modes correspond to those available in the Treatment and Service Modes. The Treatment imaging modes are also described in the “TrueBeam Technical Reference Guide Volume 2: Imaging” document. See also the section “Imaging Modes” below.

<CalibrationSet>DefaultCalibrationSetId</CalibrationSet> means that the default calibration set will be used to acquire this image.

<KiloVolts>80</KiloVolts> , <MilliAmperes>50</MilliAmperes> and <MilliSeconds>10</MilliSeconds> are the Kilo Voltage, current (in mA) and duration (in milli-seconds) of the kV image exposure pulse.

The second kV image:

This image is acquired at <Cp>2.5</Cp> which corresponds to the point where MU reaches the 120 MU point.

Acquisition parameters differ from the first kV image. Here a pulse of 70 kilo-Volts, 15 mA and 10 milliseconds in duration is used to pulse the kV source.

The second MV image:

This image is acquired at control point 4.279 which corresponds to an MU of 172.79 MU.

Imaging Modes

As shown in the previous example, the *id* attribute of the *ImageMode* element is used to select the basic imaging mode for a particular image, or sequence of images. Of course, in Developer Mode, a single XML Beam may combine acquisition of many different images, each image, or image sequence, using a different imaging mode. The *id* attribute strings that must be specified to select the various imaging modes are shown in the following table. A brief description of the basic imaging mode is also shown:

KV Imaging Modes					
Imaging Mode	String (id=)	Acquisition Mode	Acquisition Technique	Resolution	Definition
Dynamic Gain	"DynamicGain"	Dynamic Gain	Single Image	1024 x 768	A basic mode intended for CBCT projections. In this mode each pixel automatically switches to a larger gain before reaching saturation, allowing a wider dynamic range.
Dynamic Gain Fluoro	"DynamicGainFluoro"		Continuous	1024 x 768	As above, but Images are continuously acquired at 11 frames per second.
Dynamic Gain Low Framerate	"DynamicGainLowFramerate"	Dynamic Gain Low Framerate	Single Image	1024 x 768	See *Dynamic Gain.*
Dynamic Gain Low Framerate Fluoro	"DynamicGainLowFramerateFluoro"		Continuous	1024 x 768	As above, except that images are continuously acquired at 7 frames per second.
High Quality	"HighQuality"	High Quality		2048 x 1536	Full pixel resolution used.
Low Dose	”LowDose”	Low Dose	Single Image	1024 x 768	2x2 pixels merged into one to achieve higher contrast, but lower resolution.
Triggered	"Triggered"		Single Image	1024 x 768	As above, but the images are triggered by respiratory gating.
Fluoro	“Fluoro”		Continuous	1024 x 768	Same as *Low Dose* but images continuously readout at 15 frames per second

MV Imaging Modes					
Imaging Mode	String (id=)	Acquisition Mode	Acquisition Technique	Resolution	Definition
Continuous	"Continuous"	Sync	Synchronized	1024 x 768	The MV imager readout is synchronized with the pause between pulses. Several lines are read at each beam pulse, until the entire image is read out.
Dosimetry	"Dosimetry"	NoSyncDRI	Unsynchronized	1024 x 768	MV imager continuously reads out accumulated dose independent of the beam pulse.
High Quality	"Highres"	High Resolution	Radshot	1024 x 768	MV imager reads out the entire frame with full resolution during beam hold.
Triggered	"Triggered"		Radshot		Same as above but image acquisition is triggered by gating device.
Low Dose	“Lowres”	Low Resolution	Radshot	512 x 384	One image acquired in higher contrast, lower resolution mode.

Imaging modes that use the above *Single Image* KV or *Radshot* MV Acquisition techniques should be used in conjunction with the <Acquisition> element, while the imaging modes that use the *Continuous* KV, *Synchronized* MV or *Unsynchronized* MV Acquisition techniques should be used in conjunction with the <AcquisitionStart> and <AcquisitionStop> elements. Continuous imaging is discussed later in this document.

Using Dummy Control points

When taking images, it is often convenient to introduce extra dummy control points in an XML beam. These dummy control points can then be used as reference points to index image acquisition.

The following example illustrates the concept. It delivers two static fields, except that motion from the first field to the next is automatic. In addition, a pair of orthogonal kV+MV images is taken before each field.

A total of four dummy control points have been introduced to facilitate indexing where images are to be taken.

This example also illustrates how an MV and a kV image can be acquired at the same MU point. Also notice how the MLC is opened to a rectangular field in order acquire the MV image and then closed to the field shape before delivering the remainder of the field MU.

```
<VarianResearchBeam SchemaVersion="1.0">
  <!--**************************************************************************
  Two static MLC fields. First delivers 110 MU (3.2 for image + 106.8)
  Second field delivers 72 MU (3.2 for image + 68.8)
  Orthogonal MV+KV images taken before each field
  **************************************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120HD</MLCModel>
    <Accs></Accs>
    <ControlPoints>
      <Cp>  <!--cp 0: Set up to initial positions, square MLC field-->
        <SubBeam><Seq>0</Seq><Name>Demo Dummy control points</Name></SubBeam>
        <Energy>6x</Energy><Mu>0</Mu><DRate>600</DRate>
        <GantryRtn>180.00</GantryRtn><CollRtn>180.0</CollRtn>
        <CouchVrt>90.0</CouchVrt><CouchLat>100.00</CouchLat>
        <CouchLng>100.00</CouchLng><CouchRtn>180.0</CouchRtn>
        <Y1>5.00</Y1><Y2>5.00</Y2><X1>6.00</X1><X2>6.00</X2>
        <Mlc>
          <ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5.000 5.000 5.000 5.000 5.000 5.000
          5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5.000 5.000 5.000 5.000 5.000 5.000
          5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
        </Mlc>
      </Cp>
      <Cp></Cp> <!--cp 1: Use dummy cp to index KV image. -->
      <Cp></Cp> <!--cp 2: Use dummy cp to index MV image. -->
      <Cp><Mu>3.2</Mu></Cp>   <!--cp 3: Deliver 3.2 MU to square field to
                                                          take MV image. -->
      <Cp> <!--cp 4: Drive MLC and jaws to field shape.-->
        <Y1>5.31</Y1><Y2>4.81</Y2><X1>7.18</X1><X2>7.18</X2>
        <Mlc>
          <ID>1</ID>
          <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.675 0.875 1.575 1.125 1.725 2.225
          2.725 3.225 3.425 1.025 3.325 2.925 0.425 1.825 1.425 0.075 0.275 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
          <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.726 0.926 1.626 0.175 0.475 0.475
          0.075 0.525 0.625 0.325 1.025 1.825 2.025 1.625 1.025 0.525 0.326 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
        </Mlc>
      </Cp>
      <Cp><Mu>110</Mu></Cp> <!--cp 5: deliver remaining 106.8 MU-->

      <Cp><!--cp 6: No MU, just move to next field and ready MLC for
                                                          square MV image-->
        <GantryRtn>90.00</GantryRtn>
        <Y1>5.00</Y1><Y2>5.00</Y2><X1>6.00</X1><X2>6.00</X2>
        <Mlc>
```

```
        <ID>1</ID>
        <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5.000 5.000 5.000 5.000 5.000 5.000
        5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
        <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5.000 5.000 5.000 5.000 5.000 5.000
        5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
      </Mlc>
    </Cp> <!--cp x: Drive MLC and jaws to 2nd field shape.-->
    <Cp></Cp> <!--cp 7: Use dummy cp to index KV image. -->
    <Cp></Cp> <!--cp 8: Use dummy cp to index MV image. -->
    <Cp><Mu>113.2</Mu></Cp>   <!--cp 9: Deliver 3.2 MU to square field
                                                to take MV image. -->
    <Cp> <!--cp 10:  Drive MLC and jaws to field shape.-->
      <Y1>5.31</Y1><Y2>4.81</Y2><X1>7.18</X1><X2>7.18</X2>
      <Mlc>
        <ID>1</ID>
        <B>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.775 2.375 2.075 2.075 0.125 0.325
        0.225 0.175 1.275 1.475 0.825 1.425 2.325 2.425 2.225 1.525 0.525 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</B>
        <A>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.826 2.426 2.126 2.126 0.025 0.825
        1.225 1.525 1.625 2.625 2.825 2.625 2.025 1.525 1.025 0.525 0.025 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</A>
      </Mlc>
    </Cp>
    <Cp><Mu>182</Mu></Cp><!--cp 11: deliver remaining 68.8 MU for
                                                          2nd field-->
  </ControlPoints>

  <!--____________________________________________________-->
  <!--  I  M  A  G  I  N  G     S  E  C  T  I  O  N   -->
  <!--____________________________________________________-->
  <ImagingParameters>
    <DuringTreatment />
    <ImagingPoints>
      <!--_____________________
      Take KV image (1st field)
      _____________________-->
      <ImagingPoint>
        <Cp>1</Cp>
        <Acquisition>
          <AcquisitionId>1</AcquisitionId>
          <AcquisitionSpecs>
            <Handshake>true</Handshake><KV>true</KV>
          </AcquisitionSpecs>
          <AcquisitionParameters>
            <ImageMode id="HighQuality" />
            <CalibrationSet>DefaultCalibrationSetId</CalibrationSet>
            <KV>
              <KiloVolts>80</KiloVolts><MilliAmperes>50</MilliAmperes>
              <MilliSeconds>10</MilliSeconds>
              <FluoroLevelControl>None</FluoroLevelControl>
            </KV>
```

```
      </AcquisitionParameters>
    </Acquisition>
  </ImagingPoint>
  <!--_____________________
  Take MV image (1st field)
  _____________________-->
  <ImagingPoint>
    <Cp>2</Cp>
    <Acquisition>
      <AcquisitionId>2</AcquisitionId>
      <AcquisitionSpecs><Handshake>true</Handshake></AcquisitionSpecs>
      <AcquisitionParameters>
        <ImageMode id="Highres" />
        <CalibrationSet>DefaultCalibrationSetId</CalibrationSet>
        <MV />
      </AcquisitionParameters>
    </Acquisition>
  </ImagingPoint>

  <!--_____________________
  Take KV image (2nd field)
  _____________________-->
  <ImagingPoint>
    <Cp>7</Cp>
    <Acquisition>
      <AcquisitionId>3</AcquisitionId>
      <AcquisitionSpecs>
        <Handshake>true</Handshake><KV>true</KV>
      </AcquisitionSpecs>
      <AcquisitionParameters>
        <ImageMode id="HighQuality" />
        <CalibrationSet>DefaultCalibrationSetId</CalibrationSet>
        <KV>
          <KiloVolts>80</KiloVolts><MilliAmperes>50</MilliAmperes>
          <MilliSeconds>10</MilliSeconds>
          <FluoroLevelControl>None</FluoroLevelControl>
        </KV>
      </AcquisitionParameters>
    </Acquisition>
  </ImagingPoint>
  <!--_____________________
  Take MV image (2nd field)
  _____________________-->
  <ImagingPoint>
    <Cp>8</Cp>
    <Acquisition>
      <AcquisitionId>4</AcquisitionId>
      <AcquisitionSpecs><Handshake>true</Handshake></AcquisitionSpecs>
      <AcquisitionParameters>
        <ImageMode id="Highres" />
        <CalibrationSet>DefaultCalibrationSetId</CalibrationSet>
        <MV />
      </AcquisitionParameters>
```

```
          </Acquisition>
        </ImagingPoint>
      </ImagingPoints>
      <ImagingTolerances />
    </ImagingParameters>
  </SetBeam>
</VarianResearchBeam>
```

Example 7: Dummy control points facilitate specifying where exactly images are to be acquired.

One imaging programming style could be to specify a dummy control point for every imaging action. So a dummy control point in <ControlPoints> is created for every imaging action specified in the </ImagingPoints>. These dummy control points act as reference handles for the imaging section and the imaging section can be written to reference only the dummy control points.

Imaging Arm Positions, Tolerances and Motions

The previous examples, did not specify positions for the imager arms. It was assumed that the imaging arms would be placed at the proper imaging positions prior to beam on.

However, it is possible to specify explicit positions for one or more of the three imaging arms. These positions must be specified through the Imaging Points elements in the Imaging Section of the XML beam. It is also possible to specify tolerances associated with these arm positions.

Finally it is possible to move one or more of the three imaging arms during the beam. However, as previously mentioned, imaging arm motions do **NOT** follow the linear segment model as other motions do.

If a beam specifies motions for one or more of the imaging arms then:

- The imaging arms do not move between control points. Then
- When the beam reaches a control point where new arm positions are specified (in general a fractional control point), then, any beam and motion on the other axes stop. Then
- The imaging arms are moved to their new positions. Once the imaging arms have reached their new positions then,
- Any beam and motion on the other axes resumes.

The following figure illustrates how imaging arms move relative to beam and other (non imaging arm) axis motions:

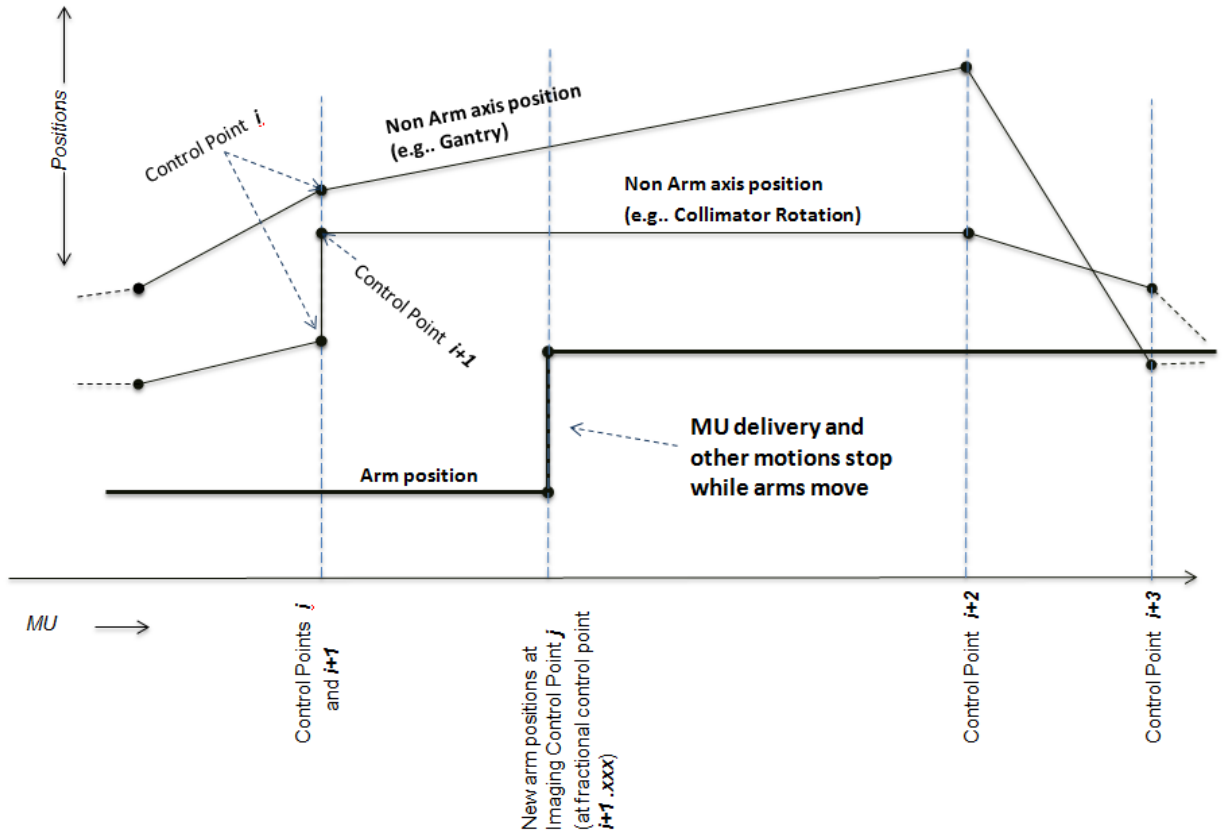


Fig. 4: Imaging Arm motions in MU vs. Position space

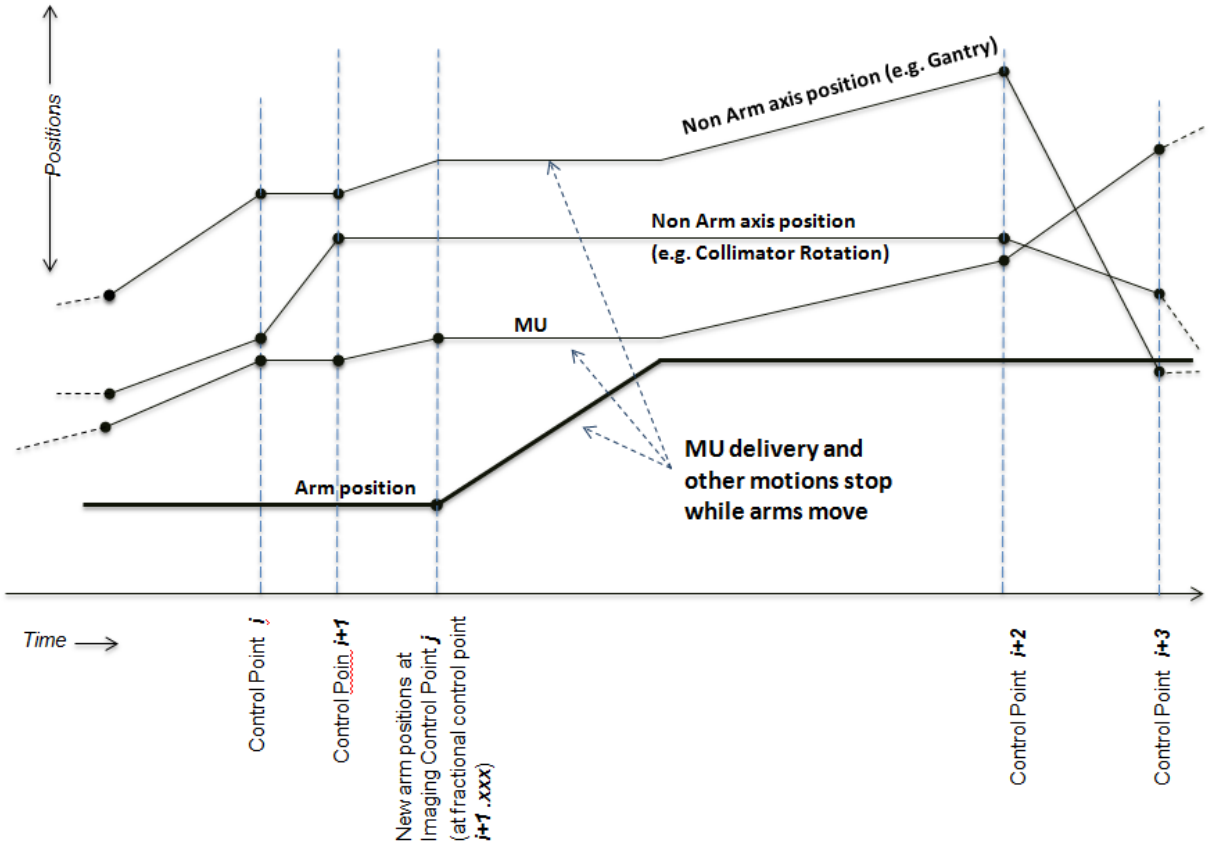


Fig. 5: Imaging Arm motions in Time vs. MU and Position space

As a corollary, it is not possible to move the imaging arms during irradiation.

The following example illustrates how to set imaging arm positions, tolerances and motions.

Xxxx Missing example: Better run this one on machine…

Show arc MV imager extended at start with tolerance. KVs come out and image for 90deg in middle of arc and then retract.

Example 8: Specifying Imaging Arm positions motions and tolerances.

Taking Continuous Images

Developer Mode also supports all the continuous imaging modes. Image acquisitions can be started and stopped at any point in the beam, using the Imaging Point elements in the Imaging Section of the XML Beam.

The following example illustrates continuous imaging. The example consists of a 180 to 0 degree arc beam where kV images are taken continuously in “DynamicGainFluoro” mode at 11 frames per second.

```
<VarianResearchBeam SchemaVersion="1.0">
<!--*****************************************************************************-->
<!--Arc Beam, delivers 180 MU over 200 degree angle (0 to 200 deg).
    5 arc regions at different MU/deg.
    Continuous KV images are taken in the 40 to 90 degree region of the arc
-->
<!--*****************************************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120HD</MLCModel>
    <Accs></Accs>
    <ControlPoints>
      <Cp> <!--control point 0 (zero).-->
        <SubBeam><Seq>0</Seq><Name>Arc N Image</Name></SubBeam>
        <Energy>6x</Energy><Mu>0</Mu><DRate>600</DRate><GantryRtn>0.00</GantryRtn>
      </Cp>
      <Cp> <Mu>40</Mu> <GantryRtn>40.00</GantryRtn> </Cp> <!--40 deg @1    MU/deg.-->
      <Cp> <Mu>100</Mu><GantryRtn>80.00</GantryRtn> </Cp> <!--40 deg @1.5  MU/deg.-->
      <Cp> <Mu>140</Mu><GantryRtn>100.00</GantryRtn></Cp> <!--20 deg @2.0  MU/deg.-->
      <Cp> <Mu>170</Mu><GantryRtn>160.00</GantryRtn></Cp> <!--60 deg @0.5  MU/deg.-->
      <Cp> <Mu>180</Mu><GantryRtn>200.00</GantryRtn></Cp> <!--40 deg @0.25 MU/deg.-->
    </ControlPoints>

    <!--This is the imaging instructions section of the beam.-->
    <ImagingParameters>
      <DuringTreatment />
      <ImagingPoints>
        <!--_____________________________________________________________________
        Make sure that KV source and imager are extended before starting
        _____________________________________________________________________-->
        <ImagingPoint>
          <Cp>0</Cp>
          <Kvd>
            <Positions>
              <Lat>0</Lat><Lng>0</Lng><Vrt>-70</Vrt><Pitch>0</Pitch>
            </Positions>
          </Kvd>
          <Kvs>
            <Positions>
              <Lat>0</Lat><Lng>0</Lng><Vrt>90</Vrt><Pitch>0</Pitch>
            </Positions>
          </Kvs>
        </ImagingPoint>

        <!--_____________________________________________________________________
```

```
        Start the KV fluoro acquisition at 40 degrees
        ____________________________________________________________________________-->
        <ImagingPoint>
          <Cp>1</Cp>
          <AcquisitionStart>
            <AcquisitionId>1</AcquisitionId>
            <AcquisitionSpecs><Handshake>true</Handshake></AcquisitionSpecs>
            <AcquisitionParameters>
              <ImageMode id="DynamicGainFluoro"/>
              <CalibrationSet>IMG-CBCT</CalibrationSet>
              <ImageDestination>ImageDestination</ImageDestination>
              <KV>
                <KiloVolts>100</KiloVolts><MilliAmperes>20</MilliAmperes>
                <MilliSeconds>20</MilliSeconds><FocalSpot>Large</FocalSpot>
                <FluoroLevelControl>None</FluoroLevelControl>
              </KV>
            </AcquisitionParameters>
          </AcquisitionStart>
          <KvFilters><Foil>1</Foil><Shape>1</Shape></KvFilters>
          <KvBlades>
            <Positions>
              <KVX1>13.3</KVX1><KVX2>13.3</KVX2><KVY1>10.0</KVY1><KVY2>10.0</KVY2>
            </Positions>
          </KvBlades>
        </ImagingPoint>
        <!--____________________________________________________________________
        Stop the KV fluoro acquisition at 90 degrees
        ____________________________________________________________________-->
        <ImagingPoint>
          <Cp>2.5</Cp>
          <AcquisitionStop>
            <AcquisitionId>1</AcquisitionId>
            <AcquisitionSpecs />
          </AcquisitionStop>
        </ImagingPoint>
      </ImagingPoints>
      <ImagingTolerances />
    </ImagingParameters>
  </SetBeam>
</VarianResearchBeam>
```

Example 9: Taking continuous images.

If a continuous image acquisition is started but never stopped then it continues until the end of the beam at which point it is turned off automatically.

The line:

```
<ImageDestination>ImageDestination</ImageDestination>
```

Causes the control system to save evey single images it takes in continuous mode.

Axis numbering

Sometimes during control system operation error messages relating to a specific mechanical axis are displayed. These axes are often referred to by numerical index.

Following is a table of the numerical index convention for axes:

Index	Axis	Index	Axis	Index	Axis
1	Gantry Rotation	2	Collimator Rotation	3	Couch Vertical
4	Couch Lateral	5	Couch Longitudinal	6	Couch Rotation
7	Collimator X1	8	Collimator X2	9	Collimator Y1
10	Collimator Y2	11	Collimator MLC	12	Reserved
13	MV Detector Shoulder	14	MV Detector Elbow	15	MV Detector Wrist
16	MV Detector Hand	17	kV Detector Shoulder	18	kV Detector Elbow
19	kV Detector Wrist	20	kV Detector Hand	21	kV Source Shoulder
22	kV Source Elbow	23	kV Source Wrist	24	kV Source Hand
25	kV X1 Collimator	26	kV X2 Collimator	27	kV Y1 Collimator
28	kV Y2 Collimator	29	kV Filter Foil	30	kV Filter Shape

Taking Images Only

It is possible to construct XML Beams that take images only. This is done by specifying a special mode the <OutsideTreatment> mode in the imaging section of an XML beam.

The following example shows how the <outside treatment> mode can be used to take kV images only. This example rotates the gantry 368 degrees and sets up the kV imager to take continuous kV images in "DynamicGainFluoro" mode (11 fps). This type of beam could be used to acquire projections for Cone Beam CT (CBCT) reconstruction. Notice that no MV radiation is delivered in this Beam.

```
<VarianResearchBeam SchemaVersion="1.0">
<!--******************************************************
Rotate gantry 368 degrees and take KV images to be used
for Cone Beam Reconstruction (CBCT).
******************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120HD</MLCModel>
    <Accs/>
    <ControlPoints>
      <Cp>
```

```
      <SubBeam><Seq>0</Seq><Name>CBCT_FullFan360</Name></SubBeam>
      <Energy>0k</Energy>
      <!--*****************************************************
      MU and dose rate are 0 since we will be taking KV images only
      and we are using the outside treatment mode
      *****************************************************-->
      <Mu>0</Mu><DRate>0</DRate>
      <!--Start the gantry in the overtravel range. -->
      <GantryRtn>364</GantryRtn>
    </Cp>
    <Cp>
      <!--Gantry rotates 368 degrees all the way on the other side
       of the overtravel range -->
      <GantryRtn>-4</GantryRtn>
    </Cp>
  </ControlPoints>

  <!--Defines the imaging parameter for the beam and each control point -->
  <ImagingParameters>
    <OutsideTreatment>
      <!--*************************************************************
      No need to set a max MU because we're taking KV images only
      *************************************************************-->
      <MaxMu>0</MaxMu>

    <!--*************************************************************
    Here is where we specify that outside treatment mode
    *************************************************************-->
    </OutsideTreatment>
    <ImagingPoints>
      <!--***********************************************
      Start continuous acquisition in DynamicGainFluoro
      mode (a good choice for CBCT)
      ***********************************************-->
      <ImagingPoint>
        <Cp>0</Cp>
        <AcquisitionStart>
          <AcquisitionId>1</AcquisitionId>
          <AcquisitionSpecs>
            <Handshake>true</Handshake>
            <KV>true</KV>
          </AcquisitionSpecs>
          <AcquisitionParameters>
            <ImageMode id="DynamicGainFluoro"/>
            <CalibrationSet>IMG-CBCT</CalibrationSet>
            <ImageDestination>ImageDestination</ImageDestination>
            <KV>
```

```
                <KiloVolts>100</KiloVolts><MilliAmperes>40</MilliAmperes>
                <MilliSeconds>10</MilliSeconds><FocalSpot>Large</FocalSpot>
                <FluoroLevelControl>None</FluoroLevelControl>
              </KV>
            </AcquisitionParameters>
          </AcquisitionStart>
          <KvFilters><Foil>1</Foil><Shape>1</Shape></KvFilters>
          <Kvd>
            <Positions>
              <Lat>0</Lat><Lng>0</Lng><Vrt>-50</Vrt><Pitch>0</Pitch>
            </Positions>
          </Kvd>
          <Kvs>
            <Positions>
              <Lat>0</Lat><Lng>0</Lng><Vrt>100</Vrt><Pitch>0</Pitch>
            </Positions>
          </Kvs>
        </ImagingPoint>

        <!--*******************************************
        Stop the continuous image acquisition...
        *******************************************-->
        <ImagingPoint>
          <Cp>1</Cp>
          <AcquisitionStop>
            <AcquisitionId>1</AcquisitionId>
            <AcquisitionSpecs />
          </AcquisitionStop>
        </ImagingPoint>
      </ImagingPoints>
      <ImagingTolerances/>
    </ImagingParameters>
  </SetBeam>
</VarianResearchBeam>
```

Example 10: Taking images only (i.e. imaging only, no therapeutic beam)

One of the main advantages of the <OutsideTreatment> mode is that MV images can be acquired without having to explicitly specify the MU required to acquire these images. The control system automatically delivers the MV MU necessary to take the images based on the image type. In general, the MU necessary to take various images can be configured in Service Mode.

The following example illustrates how MV images can be acquired in the <outside treatment> mode leaving the number of MU unspecified and thus letting the control system deliver the minimum number of MU required for the requested images.

```
<VarianResearchBeam SchemaVersion="1.0">
```

```
<!--************************************************************
Takes MV image in outside treatment mode, thus leaving the MU
unspecified. Min MU required for image requested will be
pulled out of configuration for the image requested
************************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120HD</MLCModel>
    <Accs />
    <ControlPoints>
      <Cp>
        <SubBeam><Seq>0</Seq><Name>MV Outside</Name></SubBeam>
        <!--************************************************************
        Desired energy is set to 6MV...
        But MU is left at 0. Control system will deliver
        just enough MU to acquire the image type we are requesting.
        ************************************************************-->
        <Energy>6x</Energy>
        <Mu>0</Mu>
        <DRate>300</DRate>
      </Cp>
      <Cp />
    </ControlPoints>

    <ImagingParameters>

      <!--************************************************************
      Here is where we specify that outside treatment mode
      ************************************************************-->
      <OutsideTreatment>
        <!--************************************************************
        This is just a limit of max MU to be delivered.
        Actual MU delivered will be just what is enough to take the MV image
        we are requesting.
        ************************************************************-->
        <MaxMu>100</MaxMu>
      </OutsideTreatment>
      <ImagingPoints>
        <ImagingPoint>
          <Cp>0</Cp>
          <Acquisition>
            <AcquisitionId>1</AcquisitionId>
            <AcquisitionSpecs />
            <AcquisitionParameters>
              <ImageMode id="Highres" />
              <CalibrationSet>DefaultCalibrationSetId</CalibrationSet>
              <MV />
```

```
            </AcquisitionParameters>
          </Acquisition>
          <Mvd>
            <Positions>
              <Lat>0</Lat><Lng>0</Lng><Vrt>-50</Vrt><Pitch>0</Pitch>
            </Positions>
          </Mvd>
        </ImagingPoint>
      </ImagingPoints>
      <ImagingTolerances />
    </ImagingParameters>

  </SetBeam>
</VarianResearchBeam>
```

Example 11: Letting the control system deliver only as much MV MU as necessary to acquire image(s)

Gating

It is possible specify XML Beams where the beam and/or imaging are gated in real time. The gating can be respiratory or, in general, some other device that provides real time gating information to the control system.

GATING THE kV Imaging beam:

Following is an example where kV imaging is gated. The example acquires images for CBCT but kV imaging stops when the gate is open. Notice that motions (*e.g.* gantry rotation in this example) is not gated. Only the kV beam is gated.

```
<VarianResearchBeam SchemaVersion="1.0">
<!--******************************************************
Rotate gantry 368 degrees and take KV images to be used
for Cone Beam Reconstruction (CBCT).
However take images only when gate is open and slow down
the gantry to take enough images for CBCT
******************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120HD</MLCModel>

    <!--******************************************************
    Slow down the Gantry to take enough images for CBCT in spite of the gate
    ******************************************************-->
    <VelTable><GantryRtn> 2.00 </GantryRtn></VelTable>
```

```
    <Accs/>
    <ControlPoints>
      <Cp>
        <SubBeam><Seq>0</Seq><Name>CBCT_FullFan360Gated</Name></SubBeam>
        <Energy>0k</Energy>
        <Mu>0</Mu><DRate>0</DRate>
        <!--Start the gantry in the overtravel range. -->
        <GantryRtn>364</GantryRtn>
      </Cp>
      <Cp>
        <!--Gantry rotates 368 degrees all the way on the other side
         of the overtravel range -->
        <GantryRtn>-4</GantryRtn>
      </Cp>
    </ControlPoints>

    <!--Defines the imaging parameter for the beam and each control point -->
    <ImagingParameters>
      <OutsideTreatment>
        <MaxMu>0</MaxMu>
      </OutsideTreatment>
      <ImagingPoints>
        <!--**********************************************
        Start continuous acquisition in DynamicGainFluoro
        mode (a good choice for CBCT)
        **********************************************-->
        <ImagingPoint>
          <Cp>0</Cp>
          <AcquisitionStart>
            <AcquisitionId>1</AcquisitionId>
            <AcquisitionSpecs>
              <Handshake>true</Handshake>
              <KV>true</KV>
            </AcquisitionSpecs>
            <AcquisitionParameters>
              <ImageMode id="DynamicGainFluoro"/>
              <CalibrationSet>IMG-CBCT</CalibrationSet>
              <ImageDestination>ImageDestination</ImageDestination>
              <KV>
                <KiloVolts>100</KiloVolts><MilliAmperes>40</MilliAmperes>
                <MilliSeconds>10</MilliSeconds><FocalSpot>Large</FocalSpot>
                <FluoroLevelControl>None</FluoroLevelControl>
              </KV>
            </AcquisitionParameters>
          </AcquisitionStart>
          <KvFilters><Foil>1</Foil><Shape>1</Shape></KvFilters>
          <Kvd>
```

```
            <Positions>
              <Lat>0</Lat><Lng>0</Lng><Vrt>-50</Vrt><Pitch>0</Pitch>
            </Positions>
          </Kvd>
          <Kvs>
            <Positions>
              <Lat>0</Lat><Lng>0</Lng><Vrt>100</Vrt><Pitch>0</Pitch>
            </Positions>
          </Kvs>
        </ImagingPoint>

        <!--*******************************************
        Stop the continuous image acquisition...
        *******************************************-->
        <ImagingPoint>
          <Cp>1</Cp>
          <AcquisitionStop>
            <AcquisitionId>1</AcquisitionId>
            <AcquisitionSpecs />
          </AcquisitionStop>
        </ImagingPoint>
      </ImagingPoints>
      <ImagingTolerances/>


      <!--*******************************************
      Here are the gating parameters...
      *******************************************-->
      <GatingParameters>
        <Filter>BreathingPhase</Filter>
        <QualityThreshold>0.0</QualityThreshold>
        <!--*******************************************
        Setting KV gating to true here makes the KV obey the gate...
        *******************************************-->
        <KVGating>true</KVGating>
        <GatingWindow>
          <Entry>125</Entry> <!--Close gate starting at 125 degrees.-->
          <Exit>230</Exit>   <!--Open gate starting at 230 degrees. -->
        </GatingWindow>
      </GatingParameters>
    </ImagingParameters>

    <!--*******************************************
    The beam hold device must also be listed here in the
    beam hold device area.
    Beam hold device with id="4" is TrueBeam's stnadard
    respiratory gate.
```

```
    ********************************************-->
    <BeamHoldDevices>
      <Dev Id="4" />
    </BeamHoldDevices>

  </SetBeam>
</VarianResearchBeam>
```

Example 12: Gated kV images

The above example uses "Breathing Phase" gating with the gate closing between 125° and 230°. Other types of gating are also possible, such as amplitude or breath hold.

For amplitude gating the following XML excerpt can be used for the gating parameters:

```
      …
      <GatingParameters>
        <Filter>BreathingAmplitude</Filter>
        <QualityThreshold>0.01</QualityThreshold>
        <GatingWindow>
          <Entry>1.0</Entry>  <!--in centimeter units.-->
          <Exit>2.0</Exit>    <!--in centimeter units.-->
        </GatingWindow>
      </GatingParameters>
      …
```

For breath hold gating the following XML excerpt can be used for the gating parameters:

```
      …
      <GatingParameters>
        <Filter>PassThrough</Filter>
        <QualityThreshold>0.0</QualityThreshold>
        <GatingWindow>
          <Axis>z</Axis>
          <Entry>10</Entry>  <!--in centimeter units.-->
          <Exit>15</Exit>    <!--in centimeter units.-->
          <FaultOnExit>false</FaultOnExit>
          <EntryDelay>0.5</EntryDelay>
        </GatingWindow>
      </GatingParameters>
      …
```

GATING THE MV Imaging beam:

As mentioned, the MV beam can also be gated.

Notice that, when the MV beam is gated, all beam progression, *i.e.* both MV beam and any ongoing motions, are suspended while the gate is open. Typically this leads to those mechanical axes that were in motion while the gate opened, overshooting their position while they are aggressively stopped (overshoot is more likely to happen on the heavier axes, such as the gantry). However, the control system maintains active control of the mechanical axes and immediately drives back the overshooting axes to their appropriate resumption position. Then, once the gate closes again, the axes are in position ready to resume the beam. On resumption, the finite acceleration of the mechanical axes is taken into consideration and any beam is ramped up proportionately as to maintain the MU vs. position relationship specified in the XML beam at all times. If, following a gate open and overshoot, the mechanical axes have not returned to their resumption position by the time the gate closes again (*e.g.* axis overshoot during a very short gate) then beam remains suspended until all the overshooting axes return to their resumption positions.

Following is an example of a gated MV beam. The example is an arc with 5 different zones of MU/degree densities. kV images are being continuously taken in the 40 to 90 degree region of the arc. The MV beam, gantry rotation and kV imaging are all gated.

```
<VarianResearchBeam SchemaVersion="1.0">
<!--*****************************************************************************-->
<!--Arc Beam, delivers 180 MU over 200 degree angle (0 to 200 deg).
    5 arc regions at different MU/deg.
    Continuous KV images are taken in the 40 to 90 degree region of the arc
    However, both the MV beam as well as the imaging KV beam are gated
-->
<!--*****************************************************************************-->
  <SetBeam>
    <Id>1</Id>
    <MLCModel>NDS120HD</MLCModel>
    <Accs></Accs>
    <ControlPoints>
      <Cp> <!--control point 0 (zero).-->
        <SubBeam><Seq>0</Seq><Name>Arc N Image</Name></SubBeam>
        <Energy>6x</Energy><Mu>0</Mu><DRate>600</DRate><GantryRtn>0.00</GantryRtn>
      </Cp>
      <Cp> <Mu>40</Mu> <GantryRtn>40.00</GantryRtn> </Cp> <!--40 deg @1    MU/deg.-->
      <Cp> <Mu>100</Mu><GantryRtn>80.00</GantryRtn> </Cp> <!--40 deg @1.5  MU/deg.-->
      <Cp> <Mu>140</Mu><GantryRtn>100.00</GantryRtn></Cp> <!--20 deg @2.0  MU/deg.-->
      <Cp> <Mu>170</Mu><GantryRtn>160.00</GantryRtn></Cp> <!--60 deg @0.5  MU/deg.-->
      <Cp> <Mu>180</Mu><GantryRtn>200.00</GantryRtn></Cp> <!--40 deg @0.25 MU/deg.-->
    </ControlPoints>

    <!--This is the imaging instructions section of the beam.-->
    <ImagingParameters>
      <DuringTreatment />
```

```
      <ImagingPoints>
        <ImagingPoint>
          <Cp>0</Cp>
          <Kvd>
            <Positions>
              <Lat>0</Lat><Lng>0</Lng><Vrt>-70</Vrt><Pitch>0</Pitch>
            </Positions>
          </Kvd>
          <Kvs>
            <Positions>
              <Lat>0</Lat><Lng>0</Lng><Vrt>90</Vrt><Pitch>0</Pitch>
            </Positions>
          </Kvs>
        </ImagingPoint>
        <!--_____________________________________________________________________
        Start the KV fluoro acquisition at 40 degrees
        _____________________________________________________________________-->
        <ImagingPoint>
          <Cp>1</Cp>
          <AcquisitionStart>
            <AcquisitionId>1</AcquisitionId>
            <AcquisitionSpecs><Handshake>true</Handshake></AcquisitionSpecs>
            <AcquisitionParameters>
              <ImageMode id="DynamicGainFluoro"/>
              <CalibrationSet>IMG-CBCT</CalibrationSet>
              <ImageDestination>ImageDestination</ImageDestination>
              <KV>
                <KiloVolts>100</KiloVolts><MilliAmperes>20</MilliAmperes>
                <MilliSeconds>20</MilliSeconds><FocalSpot>Large</FocalSpot>
                <FluoroLevelControl>None</FluoroLevelControl>
              </KV>
            </AcquisitionParameters>
          </AcquisitionStart>
          <!--Comment out foils and KV collimation for now...
          <KvFilters><Foil>1</Foil><Shape>1</Shape></KvFilters>
          <KvBlades>
            <Positions>
              <KVX1>13.3</KVX1><KVX2>13.3</KVX2><KVY1>10.0</KVY1><KVY2>10.0</KVY2>
            </Positions>
          </KvBlades>
          -->
        </ImagingPoint>
        <!--_____________________________________________________________
        Stop the KV fluoro acquisition at 90 degrees
        _____________________________________________________________-->
        <ImagingPoint>
          <Cp>2.5</Cp>
```

```
          <AcquisitionStop>
            <AcquisitionId>1</AcquisitionId>
            <AcquisitionSpecs />
          </AcquisitionStop>
        </ImagingPoint>
      </ImagingPoints>
      <ImagingTolerances />
      <!--*******************************************
      Here are the gating parameters...
      *******************************************-->
      <GatingParameters>
        <Filter>BreathingPhase</Filter>
        <QualityThreshold>0.0</QualityThreshold>
        <!--*******************************************
        Setting KV gating to true here makes the KV obey the gate...
        *******************************************-->
        <KVGating>true</KVGating>
        <GatingWindow>
          <Entry>125</Entry>
          <Exit>230</Exit>
        </GatingWindow>
      </GatingParameters>
    </ImagingParameters>

    <!--*******************************************
    The beam hold device must also be listed here in the
    beam hold device area.
    Beam hold device with id="4" is TrueBeam's stnadard
    respiratory gate.
    Establishing a gating device here, makes the MV beam
    obey the gate.
    *******************************************-->
    <BeamHoldDevices>
      <Dev Id="4" />
    </BeamHoldDevices>
  </SetBeam>
</VarianResearchBeam>
```

Example 12: Gating the MV beam.

Both the kV and MV beams can be gated in the same beam, however the gating parameters (gating window and gating thresholds) are common to both beams.

Validating an XML beam

For an XML Beam to be valid, it must conform to the TrueBeam XML Beam schema, i.e. it must be syntactically valid. It is typically convenient to check the syntactic validity of beams off-line, before attempting to load them into Developer Mode, as the TrueBeam accelerator is typically not available for experimentation at all times.

There are many simple commercially available XML editors and a host of other programs, such as Microsoft Excethat can be used to do this validation off-line on most PCs. While simple syntactic validation against the schema does not guarantee that the XML beam will be acceptable to the accelerator, the chance of simple syntax errors is notheless reduced. Many errors, such as, for example, the position of an axis specified in an XML beam being outside of the mechanical range, cannot be detected by simple schema validation, and thus the accelerator control system may still find further errors on an XML beam that has otherwise passed simple schema validation.

Loading and delivering an XML Beam

This section details how an XML beam is loaded into the TrueBeam control system and delivered (*i.e.* step 3 in figure 6).

To launch Developer Mode, select the “Research” option from the Major Mode Selection menu. Then type the appropriate User ID and Password.

Once in Developer Mode, you can load an XML Beam using the “Load Plan” button and selecting “Open Custom Beam” from the menu. A standard Windows file selection window will appear. Navigate to the folder containing the XML Beam you want to deliver and select “Open”. The XML Beam is now read into the TrueBeam control system. If the XML Beam is found to be invalid, an error will be displayed at this point.

Press the “PREPARE” button on the TrueBeam Control Console to prepare the beam for delivery. It is typically convenient to push the “Axis Positions” button at this point, to facilitate positioning the axes to their starting positions, as specified in the XML Beam. The initial positions required by the plan will appear in the “Program” columns of the interface. Pressing “Go To” causes the initial axes positions from the XML Plan to become Target Positions. Target positions can now be activated in groups (Gantry & Collimator group, Couch Linear group and Couch Rotation group) and motion can be initiated using either the TrueBeam control console or the in-room pendants. The process is similar to activating Target Positions and moving axes in Treatment and Service Modes.

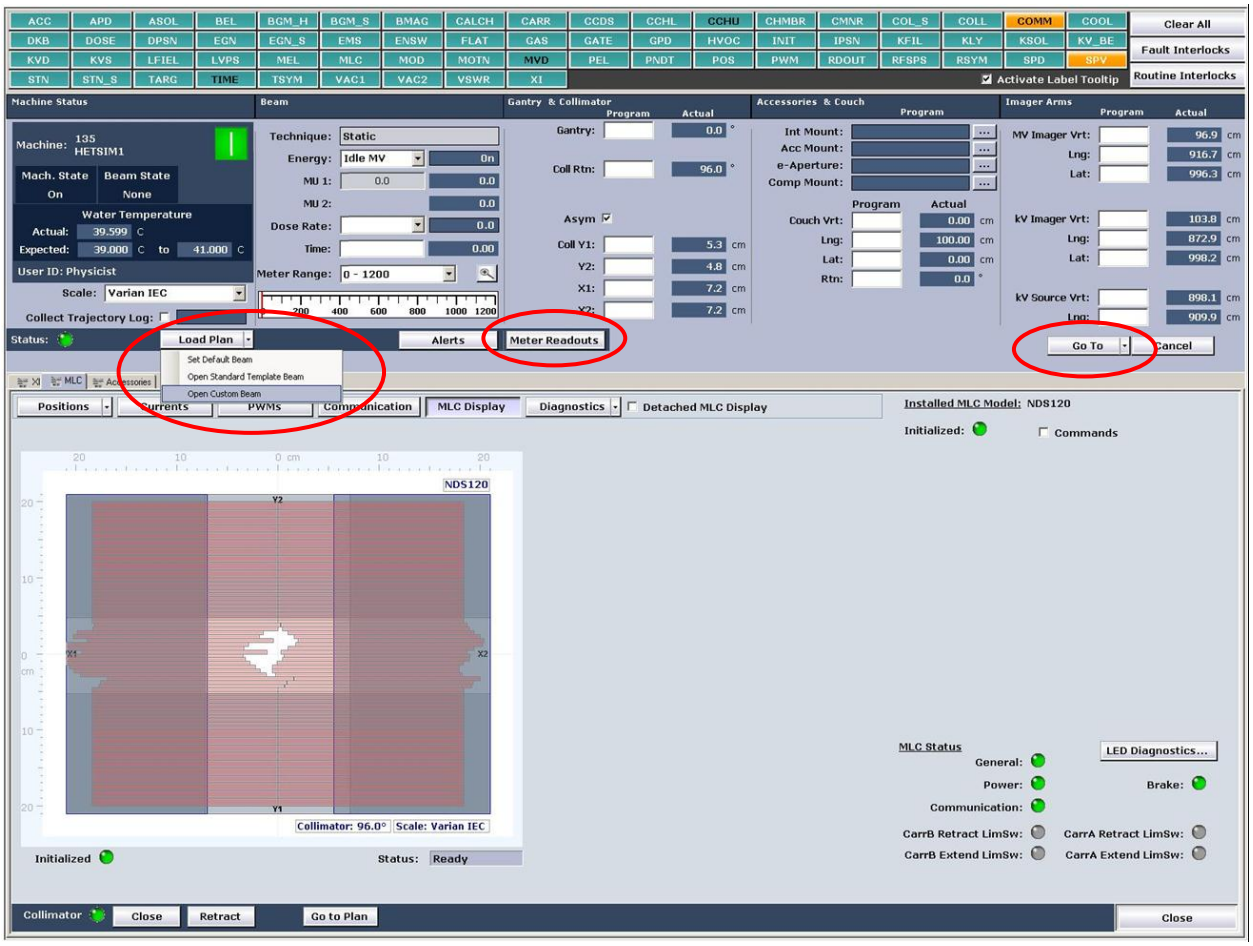
Note and Workaround: if your XML Beam requires either kV imaging foils and/or kV imaging filters, or if the XML Beam requires specific positions for the kV collimators, then you must use the trick described in the section “Indirectly Setting Up the kV Foils, kV Filters and kV Collimators” to activate these settings, since Developer Mode is currently unable to directly activate these settings prior to beam on. However, the control system still enforces these settings and will thus prevent Beam-On (through an “Initial

Position Interlock") until the kV foils, kV filters and kV blades are set up to match any such settings requested in the XML Beam.

If the XML Beam specifies any collimator accessories, these accessories would have to mounted on the machine at this point. Otherwise an "Accessory" interlock will ultimately prevent beam-on.

Once all the axes have been positioned to their starting position the "MV Ready" button will light up on the Control Console. Proceeding to beam on is the same as for the Treatment Modes: Press and hold the "MV Ready" button for a few seconds. The system will go through a pre-beam-on check cycle, then the "MV Beam On" button will light. Pressing the "MV Beam On" button at this point initiates delivery of the XML Beam.

If the XML beam that was loaded contains only kV beam ("outside treatment") then only the "kV Beam on" button will light on the Control Console.



Limitations for Imaging Calibration

Imaging calibrations for CBCT acquisitions are limited to what is also available in the Treatment Modes. The reason is because calibrations are needed for reconstruction, and the only way to calibrate a CBCT mode is using the clinical system.

Activating the Gating Hardware

If your beam uses the gating hardware you must activate it before beam on. After you have pressed the "PREPARE" button on the TrueBeam Control Console , start the NDI camera acquisition manually by extending the tracking pane as shown in the image below.

NOTE: Tracking is not activated until you have pressed the "PREPARE" button on the TrueBeam Control Console.

Typically, approximately four tracking cycles are required to establish periodicity of the tracking signal (QualityThreshold). Once periodicity is established, the Analysis section on the display changes from orange into dark blue.

Overwriting Interlocks

With the appropriate user rights, it is possible for a developer mode user to overwrite machine interlocks. User rights in developer mode follow the same general principle as Service Mode.

In general, it is not recommended to overwrite interlocks in Developer Mode. It is expected that a functional machine will be able to deliver any valid XML beam without faults and interlocks. In that sense, it is recommended that users whose access to the TrueBeam machine is limited to Developer Mode be given "Basic" user rights in the OSP (Oncology System Platform) system. Developer Mode, like Service Mode, does not allow users with "Basic" rights to overwrite interlocks and faults.

Using a DICOM file as a starting point

It is possible to use the TrueBeam Treatment mode to convert a DICOM field (or even a sequence of fields) into a Beam XML. To perform the conversion use the load file tab to load a DICOM file and then save it back as an XML.

Unfortunately, this conversion only works for DICOM files that are deliverable in the clinical modes. In other words, the validity checks performed in Developer Mode when a DICOM file is loaded, are as strict as the checks performed by the Clinical/Treatment modes.

NOTE: Any gating parameters used in the DICOM file will not be included in the LastSetBeam.xml that is generated using the procedure above.

Trajectory Records (Trajectory Logs)

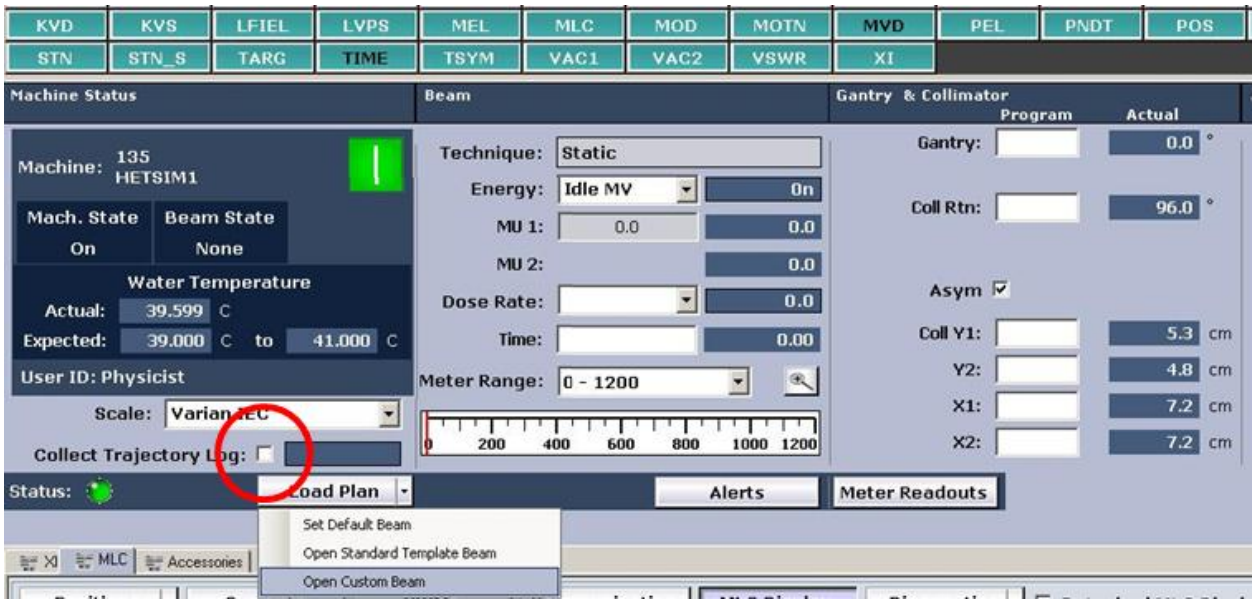
The TrueBeam control system employs many algorithms and active servo control mechanisms to ensure that the actual MU vs. position trajectory delivered matches the linear segmented trajectory specified by the user in the XML file (or other planning entity if in Treatment Mode). Generally, the control system will actively compute and manage MU delivery, velocity of moving parts, acceleration of moving parts and segment transitions, to match the actual trajectory with the planned segmented trajectory. So, for example, if MU delivery were to temporarily slow down somewhat, due to unexpected transient phenomena, the control system will proportionately slow down all moving axes so that beam progression continues along the planned segmented trajectory. When a beam first starts, (or every time the respiratory gate closes and beam resumes, if the beam is gated) the beam and motion of every moving axis are ramped up according to a pre-calculated acceleration profile. The objective is to accommodate the moving axis exhibiting the most demanding instantaneous inertia characteristics.

If motion on a moving axis stops or is not responding to motion, velocity or acceleration commands as expected by its motion model, then beam delivery is halted and a fault is generated.

In general, during beam delivery (including beamless segments) the control system continuously logs the radiation delivered and position of every moving axis every 10 msec or 20 msec (time interval depends on the control system version). The developer mode user can request that the these MU/position records be permanently stored on disk for later analysis.

Enabling Trajectory Logs

The developer mode user can request that a trajectory log be stored on disk for every beam delivered via Developer Mode. Trajectory logs are enabled by checking the box shown below on the Developer Mode user interface.



The box is not checked by default so it must be checked upon entering Developer Mode if trajectory logging is desired. While the box remains checked, the control system will generate a separate trajectory log for every beam delivered. A message to the user is displayed every time a trajectory log is generated, indicating the file name of the trajectory log. The file name contains the date and time that the Trajectory Log was generated. Trajectory logs are stored in the directory: C:\VMSOS\AppData\TDS\Output\TrajectoryLog\Research

Trajectory logs are currently stored in binary format. The exact format of trajectory record files is described in a separate document.

Trajectory logs can be large[7] and thus can potentially end up occupying a significant amount of disk space. It is recommended that trajectory logs be copied to another location at regular time intervals (*e.g.* every day, or after about a couple of hours of beam time) and then removed from the control system console computer.

Limitations

Interrupted Beams: Interrupted dynamic XML Beams pose a challenge in the current version of Developer Mode. If delivery of a dynamic Beam is interrupted because of a beam off or fault, then any moving axes will typically stop some small distance away from their required resumption point (overshoot). The amount of overshoot depends on the axis inertia. If the overshoot is greater than the internal axis tolerance then, an interlock (axis position interlock) will prevent immediate resumption of Beam delivery. Unfortunately, there is no indication on the Developer Mode interface as to what those

[7] On version 1.5 and 1.6 control systems trajectory logs require about one GB of disk space for every hour of recorded beam time.

resumption positions are and there is also no automatic way to instruct the system to move to those resumption positions. The end result is that, typically, in this situation, the XML beam must be reloaded and restarted from the beginning. This is especially true if any axis is intentionally moved by the user even further away from the resumption point.

Notice how control system response to an interrupted beam (whether interrupted due to interlock, fault or beam-off button) is different than interruption due to gating. In gating, when the gate opens the control system actively drives all moving axes (back) to the appropriate resumption point. In beams interrupted by interlock, fault or beam-off button the control system simply stops all moving axes in the shortest possible distance. That typically results in a (small) position overshoot relative to the point where the beam was interrupted.

Partial Beams (not supported): Partial Beams are not supported in the current version of Developer Mode. In the clinical Treatment Modes it is possible to instruct the system to deliver only the last X MU of a beam initially intended to deliver a total of Y MU ($X \leq Y$). The purpose of this feature in the Treatment Modes is to address the need of delivering, at a later time, the remainder of beams that had to be abandoned before completion. However, this feature is not supported in the current version of Developer Mode.

Dry Runs (not supported): The TrueBeam clinical Treatment modes support the concept of a "Dry Run" *i.e.* the ability to simulate all the dynamic motions of a Beam without actually delivering any MU [8]. However the current version of Developer Mode does not support Dry Runs.

CBCT Reconstructor not available in Developer Mode: The CBCT reconstructor that is typically available in Treatment Modes, is not currently connected to Developer Mode. Therefore, while XML Beams delivered through Developer Mode may acquire CBCT projections in any one of the imaging modes available in the Treatment modes, reconstruction must be performed through other means.

Maximum number of control points is 5000: The maximum number of control points that can be specified in the current version of Developer Mode is 5000.

Images taken by an XML beam may not be available in real time in their designated directories while the XML beam is still executing. The images typically become available a fraction of a second after delivery of the XML Beam completes.

Motion-only segments may exhibit unsynchronized motion: If a motion-only segment moves more than one axis, then the implied linearity between motions (i.e. all axes move proportionately) is maintained, unless the segment also moves the MLC. If a Motion only segment delivers neither MV nor KV radiation and there is MLC motion in the segment then the motions are not synchronized. In this non synchronized mode of motion, the axes do not follow the proportional linear motion implied in the general model and thus each axis moves independently, typically at maximum speed. In this

[8] In Treatment Modes, these Dry Runs can be performed inside the treatment room using the in-room pendant, since the primary use of this feature is to visually check equipment and patient clearances with respect to any dynamic motions to be performed during the treatment. Note that while here we are describing clinical mode features, Developer Mode is not intended for clinical use. As a reminder: Developer Mode is intended for non-clinical use only and is NOT cleared for use on humans.

uncoordinated motion case, the motion path is unpredictable as each axis is simultaneously but independently moved between the positions specified at the two control points that define the segment.

Motion-only segments trigger instantaneous stops: When the control system transitions from a beam segment (a segment delivering MU) to a motion only segment and vice versa, the control system performs a synchronized deceleration of all moving axes and MU delivery until the motions and beam come to an instantaneous complete stop. Then motions and beam are immediately re-accelerated to the speeds and dose rate required in the next segment. While this behavior does not affect the dosimetry of the beam (since positions and beam are always accelerated and decelerated in a precisely synchronized fashion) the overall time required to deliver the beam may be extended if the beam contains a large number of beam to motion only segment transitions. The total beam time extension amounts to a small fraction of a second for each beam to motion only segment transition and vice versa.

Tracking Suite

The tracking suite is a new feature available in Developer Mode 2.0 with Tracking license. It is a platform where developers can explore, test and research real-time moving anatomy tracking on the TrueBeam accelerator platform. The tracking suite enables real time compensatory motions on the TrueBeam accelerator during beam on. The general objective of such motions is to counteract the generally undesirable effect of anatomical motions in real time. In broad terms, the tracking function consists of first locating the real time position of the target and then counteracting the motion using either couch compensatory or MLC compensatory movements.

The Version 2.0 tracking suite supports only linear displacement tracking. The target volume is treated as a rigid body. Target volume rotations and deformations are not supported in this version (except perhaps magnifications due to changing proximity to the radiation source which are de-facto compensated for in couch tracking).

There are two possible position inputs to the tracking function: Calypso beacons or the optical marker block. The dDeveloper Mode 2.0 TL tracking suitemay be accompanied by the Calypso Real time target volume monitoring system. This system uses small radio transponders (called beacon transponders) to monitor both external and internal anatomical motions. The standard commercial Varian Optical Gating system ,which is used for gating the beam or position monitoring in the clinical modes, can also be used in Developer Mode to monitor real time target volume position displacements, and feed these displacements to the tracking mechanism.

The tracking suite supports both couch tracking and collimation tracking. In couch tracking, the patient support table is automatically moved to compensate for anatomical motions in real time. In collimation tracking, the instantaneous MLC treatment field is modified (shifted) to compensate for anatomical motions in real time. Therefore, for static collimation fields the MLC simply moves to compensate for the target's motion in the beam's eye view. For dynamic fields (e.g. IMRT, RapidArc or other experimental dynamic techniques) the tracking motion is superimposed onto the plan's dynamic

motion. In other words, in collimator tracking, the inherent dynamic collimation of dynamic plans is further modulated by the tracking compensatory motion in real time. Similarly in couch tracking of plans with inherent pre-planned couch motions, the planned couch motion is combined with the tracking couch motion. What about jaws?

In collimator tracking the target volume is treated as a rigid volume. The 3D displacement of the target volume is mapped into the beam's eye view and then a collimator opening displacement is applied, i.e. modification of the MLC shape. Rotations and non-rigid deformations of the target volume (including magnification due to varying distance from the radiation source) are not supported.

Developer Mode 2.0 provides a graphical user interface (GUI) to program and control the tracking suite. This GUI encapsulates many of the underlying tracking complexities. The tracking suite allows developers to set up a tracking experiment and also allows for the retrospective analysis of past experiments (review mode). The review mode can also run separately as a standalone program, outside Developer Mode 2.0.

In summary, the tracking suite supports the following main operations:

- Create a tracking test (XML), load a previous tracking test, or load a template,
- Run tests and display results in real time,
- Record tracking data and any acquired images,
- Allow for review and evaluation of a previous run,
- Off line review mode (separate module)

The XML used to describe beams in Developer Mode has been expanded in version 2.0 to include tracking information. Like the treatment and imaging parameters, all the tracking parameters are contained in the XML Beam itself.

In general, those tracking parameters that affect the function and performance of tracking are part of the XML Beam specification itself. Those parameters that determine what will be displayed on the tracking suite GUI during a tracking experiment (e.g. the tracking traces that will be shown in real time, whether a tracking experiment's data will be saved and where) are specified in the GUI, not the XML Beam script.

Simple tracking cases can be set up using the GUI or by loading the available predefined tracking setup templates. In these cases, the UI adds the appropriate tracking parameters to the XML Beam. More advanced use cases may require explicit editing of the XML Beam.

Tracking Concepts

In order to compose XML beams that activate the tracking functionality of TrueBeam 2.0, it is necessary to become familiar with some tracking concepts. These concepts are described in the next sections. These concepts are also part of the Version 2.0 XML Beam schema.

Enabling Tracking

A Set Beam XML script enables tracking through the "Tracking" element. Presence of this element enables the tracking functionality for the specific XML beam only. The "Tracking" element is used to specify:

a) which axes will participate in the tracking (e.g. Couch and/or MLC) and other details
b) a conformity tolerance specifying the acceptable fluence of overexposure and underexposure (read "Conformity Index" section later on for details) and
c) the "InitialCapabilityRatio" specifying how much reserve velocity should be kept in reserve for tracking (read "Capability Ratio" section later on).

Enabling tracking is subject to the following rules:

- For couch tracking, the axis tolerance or the conformity tolerance must be set for each couch tracking axis.
- Jaw tracking requires that MLC tracking also be enabled.
- MLC tracking requires setting of a conformity tolerance (see "Conformity Index Tolerance") section below
- All tracking axes must have at least initial positions listed in the first control point, even if those axes are otherwise static axes as far as the plan is concerned.
- If conformity tolerance is specified, there must be at least initial positions for the MLC and collimator jaws (perhaps put this bullet above the prior one, so that all MLC tracking poins are together)

Tracking Axis List

The "TrackingAxisList" element is used to specify the mechanical axes that will participate in tracking. Some expected choices are couch tracking and collimation tracking. Couch tracking can be limited to one axis (e.g. couch longitudinal only) or can involve up to all three of the translational couch axes (vertical, lateral, longitudinal). Tracking of rotations is not supported in this version, therefore couch rotation, pitch and roll cannot participate in tracking at this time.

MLC tracking is another choice. If MLC tracking is enabled, Y and/or X collimator jaw tracking can also be enabled to track the most retracted MLC leaves and first closed leaf pairs (see also discussion about "circulating leaves" for more details).

The tracking suite also supports a different type of tracking: "Phase Tracking". In this mode, every control point is tagged with phase information, and the instantaneous beam delivery speed is adjusted to track the phase. See "Phase Tracking" section later on for more details.

Circulating MLC leaves

Target motions parallel to the direction of the MLC leaf travel are generally easier to compensate for than target motions perpendicular to the leaf travel. When the target moves perpendicular to the direction of the MLC leaves, new leaf pairs must open and close. If the closed leaf pairs were to be kept

under the collimator jaws (as is typically done in non-tracking fields) then it would take too long to bring new leaf pairs out from under the jaws, delaying treatment delivery and complicating tracking.

To address these perpendicular target motions, the tracking suite keeps a few pairs of closed leaves close to the target, not under the jaws. Because closed leaf pairs involve increased leakage in the region of their touching edges, the tracking suite keeps "circulating", i.e. moving back and forth, these closed leaf pairs to avoid the creation of undesirable hot spots in the field. For targets that are expected to move quickly, more closed leaf pairs are kept in reserve on each side of the moving target volume. Conversely, for targets that are expected to move slowly, fewer leaf pairs are kept in reserve.

A few more parameters determine the behavior of circulating leaf pairs in MLC tracking:

"ExpectedTargetSpeed": used by the tracking suite to determine how many closed leaf pairs to keep in reserve, along the X direction, to address target motion perpendicular to the leaves. Specifying a value >0 results in at least one, or more, closed leaf pairs being kept in reserve past the X dimension of the instantaneous radiation field.

"YTargetRange": used by the tracking suite to bound the Y direction travel of circulating leaves.

MLC Carriages during Tracking

In non-tracked fields, the MLC carriages are typically driven as tightly as possible around the MLC shape (or the CIAO for IMRT and RapidArc® type fields). With tracking, the carriages must be retracted a little to avoid the need for carriage movements during tracking (moving the carriages would require a beam hold which would delay delivery and also complicate tracking).

It is possible to control how far back to hold the carriages from the MLC shape (or CIAO), by setting the "OpenUpCarriages" element. The units are in centimeters.

Target vs. Baseline Tracking

Each tracking axis, whether MLC, jaws or couch, can be set up for either "target" or "baseline" tracking. In baseline tracking, only baseline shifts are compensated, while in target tracking the entire instantaneous displacement is tracked.

Combination tracking with some axes doing baseline tracking while others doing target tracking are also possible. One possible choice may be baseline tracking assigned to the couch and target tracking assigned to the collimation (MLC and possibly collimator jaws).

The Conformity Index

The tracking suite also implements a metric intended to summarize how successful (how precisely) the system is in tracking a moving target. A number of factors may limit the accuracy of tracking, the three main challenges being: (a) the maximum velocity of the couch top and MLC are finite, (b) the finite MLC leaf thickness implies a discretization that contributes to tracking discrepancies and (c) pre-planned dynamic couch or MLC motions (e.g. IMRT and RapidArc®), inherent in the plan itself, limit the speed margin available for compensatory tracking motions.

The conformity index is an area metric that measures the discrepancy between the current instantaneous collimation fluence area vs. the desired fluence area were the target volume to be perfectly tracked. The conformity index is a two number metric, measuring separately the overdose and underdose areas. The two overdose and underdose areas are measured in square centimeters (cm^2). An instantaneous percentage value is also calculated and displayed on the GUI showing the ratio between the overdose and underdose areas normalized by the instantaneous area of the total fluence (i.e. normalized by the total instantaneous field size).

The conformity index can be used to trigger certain real time actions, such as suspending (holding) the beam if the conformity index deteriorates past a value selected in the XML Beam.

Conformity Index Tolerance

Tracking often requires the specification of a conformity index tolerance (see rules in "Enabling Tracking" section above). This tolerance can be used to specify when certain actions must be performed, such as temporary suspension of the beam or temporary suspension of tracking motions. As mentioned above, the conformity index is a two number metric quantifying separately under exposure and over-exposure. Therefore a separate "OverExposure" tolerance and a separate "UnderExposure" tolerance are needed to specify the conformity tolerance.

The Capability Ratio

As mentioned earlier, dynamic treatments that contain couch motion or MLC motion (e.g. IMRT and RapidArc) will, in general, limit the available max velocity margin available for compensatory tracking motions. This is because the instantaneous required compensatory motion may combine with the inherent plan motion and exceed the maximum velocity of one or more MLC leaves or the couch.

Normally, absent tracking, the speed trajectories of moving axes in a dynamic treatment are calculated so that either the speed of a moving axis (e.g. the fastest moving MLC leaf or the fastest moving couch axis) or the dose rate is maximized. This is done to enable delivery of beams in the shortest possible time. With tracking, it is useful to limit the maximum planned speed of axes, so that some velocity margin is left available for compensatory tracking motions. This is done by specifying an "InitialCapabilityRatio" value in the XML Beam. The valid values are $0 < \text{InitialCapabilityRatio} \leq 1.0$. A value of 1.0 means that the plan will be delivered at the maximum possible speed (of course, this leaves zero margin for tracking whenever a tracking axis is already moving at its max speed and the required tracking motion is in the same direction as the planned motion). On the other hand, very small values for the capability ratio will significantly lengthen beam delivery time.

Given the interplay between planned motions inherent in the plan and compensatory tracking motions, a lower capability ratio should be specified with fast moving targets and vice-versa.

Motion Management Parameters

The remainder of the tracking parameters have been grouped under the "MotionManagementParameters" element. This is because tracking is obviously a motion management technique, like gating, intended to address the otherwise undesirable effect of unplanned anatomy movements.

Tracking sources

The Developer Mode 2.0 tracking suite supports two tracking sources to guide the motion compensation.

(a) The NDI camera
(b) The RTX interface

The NDI camera is the same system that is used to track an optical marker block on a patient's surface in the commercial Varan Optical gating system. In the V2.0 tracking suite this NDI camera can be used to track continuous movement, except that the intention is to track (as opposed to simply gating the beam).

The tracking source must be specified as a string (either "NDI" or "RTX1") in the "id" attribute of the "TrackingSource" element.

Each tracking source is also accompanied by a Surrogate Model, a Motion Model and Tracking Action Windows. These concepts are described below.

The Surrogate Model

Many systems designed to monitor and track internal anatomical motions, directly measure movements in the target volume, or very close approximations to those movements. Such a system is the Calypso Real time target volume monitoring system, when used with internally implanted radio transponders. If those radio transponders are implanted in close anatomical proximity to the target volume, then the observed motion of the transponders closely approximates motions of the target volume itself.

However, in the more general case, a real time anatomical movement system only measures indirect movement ("surrogate" movements), which then must be mapped to internal anatomical movement. Such a system is the standard commercial Real Time Position Management (RPM) system. This system is typically used to monitor and simply gate the beam, but the 2.0 tracking suite also allows its use as a tracking device to monitor real time movement. The marker block is typically placed somewhere on a patient's skin (typically the abdomen or thorax) to monitor respiratory movements. These types of systems measure primarily vertical abdominal motion in the Anterior Posterior direction (Z axis). When such a system is used for tracking, the marker block movement must be transformed (mapped) into the presumed internal anatomical Superior-Inferior motion (Y axis) that is typical of respiration. The motion of the markeblock is essentially used as a "surrogate" motion for the motion of the internal anatomy.

The device whose movement is being monitored is generally referred to as "the surrogate", and its observed motion as the as the "observed surrogate motion" or simply "surrogate motion".

The Calypso Real time target volume monitoring system, when used with surface radio transponders is also a system that only indirectly measures internal target motion. Even when internal implanted transponders are used, the observed motions do not exactly match the target volume movements and thus minor corrections may be applicable.

With those systems that only measure surrogate motions, a transform becomes necessary to establish how the measured motion correlates to the internal target motion, i.e. how the measured motion

correlates to internal anatomy target motion. This is the "surrogate" transform. A typical surrogate transform using the NDI camera might be to map the observed Z motion of the marker block into Y respiratory motion.

The tracking suite supports two types of surrogate transforms:

a) the "Basic" surrogate model which is essentially the identity surrogate model and

b) the "Amplitude Fit" surrogate model

One of the two models must be chosen. Details for each model are given below.

The Basic Surrogate Model

The basic model is a fixed (rigid) transform where the internal anatomical target motion is assumed to be the same as the motion of the observed surrogate device. The basic surrogate model is operationally the simplest transform. This model may be a good approximation for internal Calypso radio transponders that are implanted in close proximity to the target volume. In this basic model the target position is assumed to be at a fixed offset from the monitoring device position, hence the motion of the target is assumed to be the same as the motion of the monitoring device.

The fixed transform of the Basic Surrogate Model can be specified in the XML Beam as a fixed transformation between two Cartesian coordinate systems. Read later on under "placement" to see the convention used to describe transforms between Cartesian coordinate systems.

Specifying a transform for the basic surrogate model is optional and if no explicit transform is given in the XML Beam then the identity transform is assumed. Otherwise, the supplied "placement" is used as a fixed displacement between the surrogate and target volume positions.

Note: The Basic Surrogate Model can be used in conjunction with the (RPM) system optical marker block positioned on a moving stage on the couch, to evaluate how well couch tracking compensation can hold the marker block in position.

The Amplitude Fit Surrogate Model

The amplitude Surrogate model maps (fits) the observed amplitude and velocity into a 3D X, Y, Z internal anatomy displacement.

Supported fit types are:

- "Static"
- "Linear"
- "Quadratic"
- "Cubic"
- "Planar"

The first three types are essentially polynomial fits.

A different fit can be specified for each one of the X, Y, Z axes. The desired fit type for each axis must be specified in the XML beam as a string in the "FitType" element.

Each fit is accompanied by up to four coefficients a_0, a_1, a_2 and a_3 . For static fit the amplitude A is mapped as a_0, for linear fit as $a_0+ a_1*A$, for quadratic fit as $a_0+a_1*A+a_2*A^2$ and for cubic as $a_0+ a_1*A + a_2*A^2+a_3*A^3$. For planar fit the amplitude A and rate V are mapped as $a_0+a_1*A+a_2*V$.

For example, if the optical marker block and NDI camera are used for tracking, and if the tracking simulates the typical Z direction motion on a patient's abdomen, then the required transform may simply map the Z motion into an internal anatomical target volume Y motion with a gain and offset. In this case the surrogate transform would be something along the lines of:

Y_{target} = Gain * $Z_{surrogate}$ – Gain * $Z_{acquired}$

Placement

In the current tracking implementation, all transforms between Cartesian coordinate systems are described as "placements". A Placement is essentially a way to describe the relative spatial position between two coordinate systems (X,Y,Z) and (X',Y',Z').

In the current implementation it was chosen to describe such transforms using four vectors. One vector indicating the location of the origin of the (X',Y',Z') system with respect to (X,Y,Z) and the other three vectors to indicate the orientation of the three (X',Y',Z')axes inside the (X,Y,Z) coordinate system. X',Y' and Z' are specified as unit vectors with a length of 1. Therefore a total of 12 numbers (there for each of the four vectors) are needed to describe a coordinate transform using this convention. This is a redundant specification and requires that the X',Y',Z' unit vectors result orthogonal to each other. If they are not, the placement transform is flagged as invalid.

The Motion Model

Sometimes the observed motion can be used "as is" to drive the compensation. However, if the motion exhibits periodicity then a more complex motion model can be used to extract the underlying periodic components of the motion and drive the tracking compensation.

The tracking suite on Developer Mode 2.0 supports two motion models:

a) The "BasicMotionModel" and

b) The "GatingMotionModel"

The Basic Motion Model

This model is typically used when the observed motion is to drive the tracking compensation, as is. This, for example, might typically be the case with prostate-like target volume movements observed with Calypso implanted radio transponders. These motions are typically regarded as erratic and unpredictable. Hence, these motions do not exhibit periodicity, amplitude, phase or quality. In these cases the "BasicMotionModel" may be used and the observed motion directly drives the tracking compensation (the specified surrogate transform is always applied, of course).

To select the Basic Motion Model specify "BasicMotionModel" in the XML Beam.

The Gating Motion Model

If the motion exhibits periodicity, for example the typical breathing motion, then the "GatingMotionModel" can be chosen to analyze the observed motion, derive macroscopic parameters, such as amplitude, phase and quality. In this case the smoothed periodic sinusoidal signal is used to drive the tracking compensation (the surrogate transform is also applied).

To select the Gating Motion Model specify "GatingMotionModel" in the xml Beam.

When "GatingMotionModel" is specified, Developer Mode V2.0 uses the same algorithms employed in the commercial Real Time Position Management (RPM) respiratory gating system to compute the values of amplitude, phase and quality, and smooth the observed motion into a periodic sinusoidal pattern. This smoothed sinusoidal motion is then used to drive the tracking compensation.

The sinusoidal motion of the "GatingMotionModel" is assumed to be along a line in 3D. By default, the motion is assumed to be along the Z axis (typically anterior-posterior), however a different direction can be specified by supplying a vector inside the GatingMotionModel element of the XML beam. The vector is specified by supplying its X,Y and Z axis components.

The motion model also extrapolates (i.e. predicts) the future position from the observed motion. This prediction is necessary because of some small but unavoidable latency, and because the moving parts used for the compensatory tracking motions (the couch and the MLC) have a finite inertia. It is therefore important to determine not only the likely position of the surrogate in the immediate future, but also its expected velocity. This prediction increases tracking accuracy and is most important for motions involving mechanical moving parts with low (acceleration) / (max velocity) ratios. Note that, in this context, the basic motion model also constitutes a prediction. It is the default prediction that whenever the surrogate moves, it will then stay at its new position and thus its velocity will also be zero in the immediate future.

The Fixed Room coordinate system

In the following sections, the term "fixed room coordinate system" is used. This term generally refers to the fixed room coordinate system where the machine isocenter exists as a fixed point at the origin, the Z axis is vertical, and the Y axis is the axis of rotation for the gantry. Motion traces can be selected for real time viewing in the GUI. The traces can be shown in the fixed room and/or the model coordinate system.

Tracking Action Windows

Action windows generally specify limits, or trigger levels, for a variable. They also specify actions (e.g. suspension of the beam) to be taken when the variable moves outside the specified limits.

For the purposes of tracking, action windows can be specified for the following variables/axes in various coordinate systems (i.e. various frames of reference). The name of the variable must be listed as a string in the "axis" attribute of the action window:

- "Amplitude" : The amplitude of the observed surrogate motion in the model system (i.e. movement relative to couch top). Units: cm

- "Phase" : The phase of the above motion in degrees [0-360]
- "Quality": The quality of the above motion

- "Target_fixed.X" : The motion of the target volume along the X axis in the fixed room coordinate system. Units: cm
- "Target_fixed.Y" : As above, but for the Y axis
- "Target_fixed.Z" : As above, but for the Z axis
- "Target_fixed.R" : The radial target motion in the fixed room coordinate system. Radial motion is defined as: $\sqrt[2]{X^2 + Y^2 + Z^2}$, i.e. the length of the X,Y,Z displacement vector. Units: cm
- "Target_fixed.AngleX" : The motion of the target volume around the X axis in the fixed room coordinate system. Units: degrees
- "Target_fixed.AngleY " : As above, but for the Y axis
- "Target_fixed.AngleZ " : As above, but for the Z axis

- "Target_model.X" : The motion of the target along the X axis in the model coordinate system (i.e. movement relative to couch top). Units: cm
- "Target_ model.Y" : As above, but for the Y axis
- "Target_ model.Z" : As above, but for the Z axis
- "Target_ model.R" : The radial target motion in the model coordinate system. Units: cm
- "Target_ model.AngleX" : The motion of the target volume around the X axis. Units: degrees
- "Target_ model.AngleY" : As above, but for the Y axis
- "Target_ model.AngleZ" : As above, but for the Z axis

Traces for these variables can also be displayed in real time on the tracking suite GUI.

Note that target motions in the fixed room system will typically contain the combined motions of pre-planned motions from the plan itself, superimposed on the motions due to any compensatory couch motion (if couch tracking is used). Target motions in the model system will contain only true target motions with respect to the couch top, and will therefore exclude pre-planned motions from the plan or couch tracking itself.

The "Lower Limit" and "Upper Limit" elements specify the action window range. The units of the range are axis specific (see above list of axes).

Action windows can also be defined as either "basic" or "segmental". In the "basic" case, the action ceases automatically when the variable re-enters the specified action window. For example, an action window on the Target_ model.Y axis specified with an MVBeamImpact causes the beam to be suspended when target volume motion relative to the couch top exceeds the specified range. As soon as the variable returns inside the specified position range *the beam resumes automatically*. If, however, the action window is specified as "segmental", then, upon window exit, the *delivery stops* and the user is prompted to approve continuation.

The "MotionCompensationImpact" element controls whether an action window suspends tracking or not. Note that a tracking window would also be typically be set up to suspend the MV beam

("MVBeamImpact" would be typically set to "true"). The MotionCompensationImpact would then be used to control whether tracking also stops.

Setting the "FaultOnExit" element to "true" causes the delivery to stop with a fault whenever the monitored variable exits the specified window range. User Beam-on is then required to resume.

The Model system

In order to be compatible with dynamic beams that contain pre-planned couch movements, the Developer Mode 2.0 tracking suite implements a "Model" Coordinate System.

For example, beams used to research non-coplanar trajectory stereotactic techniques, where the treatment plan requires the coordinated synchronization of couch rotation and gantry rotation (to achieve arbitrary, non-coplanar trajectories for the incident beam direction) are examples of XML beams with pre-planned couch motions. Another example may be research into total body, or very large field irradiation, where the couch longitudinal motion (either step-and-shoot or continuous) is used to irradiate large fields. In these cases, any compensatory tracking couch motion is overlaid on the pre-planned couch motions.

Both target volume movements as well as couch movements contained in the plan itself will cause a change in observed surrogate position. In order to properly support such beam deliveries, the tracking suite must be able to distinguish between (a) observed movement due to couch motion inherent in the plan and (b) true motion due to (presumably) moving anatomy. Obviously, observed motions attributable to pre-planned couch movement must be ignored for the purposes of calculating compensatory tracking motion.

Internally, the Developer Mode 2.0 tracking suite accomplishes this motion separation by defining a "Model Coordinate System". This system is always pinned to the couch top, and all observed motions are mapped into the Model Coordinate System. If the target volume moves with the couch (as would happen for pre-planned couch motions inherent in the plan), then the net target volume position in the model system remains the same, and no compensation is necessary. If the target volume has also moved with respect to the couch top, then this component of the observed motion will persist even after translation into the model coordinate system, and must be compensated for by tracking.

As a note, in the case of couch tracking, the existence of the model system also facilitates distinction between observed motions attributable to anatomy movement, vs. observed motions due to compensatory couch motion itself. As a hypothetical example, in an ideal case where couch tracking is perfect, observed motion would be zero, while motion in the model system would be the inverse of the compensatory couch motion, the two motions cancelling each other completely in the fixed room / isocenter coordinate system. In practice, observed motion will equal the imperfections of couch tracking, showing residual uncompensated motion.

By default, the model system is pinned to the defined couch position in the first control point of the XML Beam. If the XML Beam does not define couch positions, then the model system is pinned to the instantaneous couch position found when the XML beam is loaded into the system.

It is also possible to define a custom Model System transform by defining a set of couch positions (Lateral, Longitudinal, Vertical, Rotation, Pitch and Roll) inside the "ModelSystem" element of the XML Beam.

Other tracking parameters

The “MotionCompensation” element is reserved primarily for future use and must currently be set to “Basic”. Tracking of target volume rotations is not supported in this version.

Phase Tracking

As mentioned, the tracking suite also supports “Phase Tracking”. Phase tracking is activated by selecting “Phase” as an axis in the TrackingAxisList (see previous section “Tracking Axis List”).

Phase Tracking requires that every control point be assigned a phase value (see the optional “phase” element inside “cp” definition). The phase is in degrees (0.0≤phase<360.0). Then, during beam-on, the instantaneous beam delivery speed is dynamically adjusted to track the phase.

Phase tracking may fall behind the actual measured phase, at which point it may be preferable to halt delivery and wait an almost complete period to resynchronize. The maximum allowed phase lag allowed before waiting for resynchronization is controlled through the “MaxPhaseLag” attribute. Units are degrees (0.0≤ MaxPhaseLag <360.0).

In summary, the objective of phase tracking is to allow pre-planned tracking of a moving target volume (i.e. pre-planned modulation of the instantaneous field size), based on a relationship between phase and displacement which is presumably known at treatment planning time. Phase tracking typically requires that the general target trajectory, as well as the machine’s speed capability in delivering a particular treatment, be both known at treatment planning time.

APPENDIX A: The Varian TrueBeam XML Schema

Following is the Varian TrueBeam XML Schema for Developer Mode 2.0:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:VMSHET="xsd.os.varian.com/HET" xmlns:xs="http://www.w3.org/2001/XMLSchema">


  <!--
  ********************************************************************************
  ***
  ***  Generic Definitions
  ***
  ********************************************************************************
  -->

  <xs:simpleType name="unsignedDouble">
    <xs:annotation>
      <xs:documentation>
        xs:double restricted zero and positive double.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:double">
      <xs:minInclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="NoneValue">
    <xs:annotation>
      <xs:documentation>
        The "none" value string.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="none" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="DoubleNone">
    <xs:annotation>
      <xs:documentation>
        xs:double extended to include the value "none".
      </xs:documentation>
    </xs:annotation>
    <xs:union memberTypes="xs:double NoneValue  " />
  </xs:simpleType>

  <xs:simpleType name="doubleRatio">
    <xs:annotation>
      <xs:documentation>
        xs:double restricted to between 0.0 to 1.0
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:double">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="1"/>
    </xs:restriction>
```

```
</xs:simpleType>

<xs:simpleType name="doublePhase">
  <xs:annotation>
    <xs:documentation>
      xs:double restricted to between [0.0 to 360.0]
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:double">
    <xs:minInclusive value="0"/>
    <xs:maxExclusive value="360"/>
  </xs:restriction>
</xs:simpleType>

<!--
  ********************************************************************************
  ***
  ***  System Wide Simple Type Definition
  ***
  ********************************************************************************
-->
<xs:simpleType name="TreatmentModeType">
  <xs:annotation>
    <xs:documentation>
      Possible TreatmentMode values defined in restrictions below.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction   base="xs:string">
    <xs:enumeration value="FTM"/>
    <xs:enumeration value="ISTM"/>
    <xs:enumeration value="QATM"/>
    <xs:enumeration value="QATM_VERIFICATION"/>
    <xs:enumeration value="MCTM"/>
  </xs:restriction>
</xs:simpleType>

<!--
********************************************************************************
***
***  Imaging Axes
***
********************************************************************************
-->

<xs:simpleType name="KvFilterType">
  <xs:annotation>
    <xs:documentation>
      kV Filter position between 0 and 2.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0" />
    <xs:maxInclusive value="2" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SpecialPositionType">
  <xs:annotation>
    <xs:documentation>
      Predefined positions for arms:
      Position Type:
```

```
      retracted - arm is in a pre-defined fully retracted position.
      extended  - arm is either in a pre-defined extended position, or in a pre-defined
                  clinical area (the Vrt and Lng coordinates are within a pre-defined
                  polygon where the image can be taken according to the treatment plan).
      mid       - arm is in a pre-defined position between retracted and extended.
      none      - all other positions.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="retracted" />
    <xs:enumeration value="mid" />
    <xs:enumeration value="extended" />
    <xs:enumeration value="none" />
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="ArmAxesType">
  <xs:annotation>
    <xs:documentation>
      Arm axes clinical positions; units in cm / deg.
    </xs:documentation>
  </xs:annotation>
  <xs:all>
    <xs:element name="Lat" type="DoubleNone" minOccurs="1" maxOccurs="1" />
    <xs:element name="Lng" type="DoubleNone" minOccurs="1" maxOccurs="1" />
    <xs:element name="Vrt" type="DoubleNone" minOccurs="1" maxOccurs="1" />
    <xs:element name="Pitch" type="DoubleNone" minOccurs="1" maxOccurs="1" />
  </xs:all>
</xs:complexType>

<xs:complexType name="KvFiltersPositionType">
  <xs:all>
    <xs:element name="Shape" type="KvFilterType" minOccurs="0" maxOccurs="1" />
    <xs:element name="Foil" type="KvFilterType" minOccurs="0" maxOccurs="1" />
  </xs:all>
</xs:complexType>

<xs:complexType name="BladePositionsType">
  <xs:annotation>
    <xs:documentation>
      Positions of the kV Blades; units in cm.
    </xs:documentation>
  </xs:annotation>
  <xs:all>
    <!--units are cm-->
    <xs:element minOccurs="1" maxOccurs="1" name="KVX1" type="DoubleNone" />
    <xs:element minOccurs="1" maxOccurs="1" name="KVX2" type="DoubleNone" />
    <xs:element minOccurs="1" maxOccurs="1" name="KVY1" type="DoubleNone" />
    <xs:element minOccurs="1" maxOccurs="1" name="KVY2" type="DoubleNone" />
  </xs:all>
</xs:complexType>

<xs:complexType name="ImagingTolerances">
  <xs:annotation>
    <xs:documentation>
      Tolerances of the imaging equipment (imaging arms, kV collimation system).
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!--units are deg or cm -->
    <xs:element minOccurs="0" maxOccurs="1" name="Mvd" type="ArmTolerances" />
```

```
    <xs:element minOccurs="0" maxOccurs="1" name="Kvd" type="ArmTolerances" />
    <xs:element minOccurs="0" maxOccurs="1" name="Kvs" type="ArmTolerances" />
    <xs:element minOccurs="0" maxOccurs="1" name="KvBlades" type="KvBladesTolerances" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="KvBladesTolerances">
  <xs:annotation>
    <xs:documentation>
      Tolerances of a kV Blades.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!--units are deg or cm-->
    <xs:element name="KVY1" type="xs:double" minOccurs="1" maxOccurs="1" />
    <xs:element name="KVY2" type="xs:double" minOccurs="1" maxOccurs="1" />
    <xs:element name="KVX1" type="xs:double" minOccurs="1" maxOccurs="1" />
    <xs:element name="KVX2" type="xs:double" minOccurs="1" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ArmTolerances">
  <xs:annotation>
    <xs:documentation>
      Tolerances of an arm.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!--units are deg or cm-->
    <xs:element name="Lat" type="xs:double" minOccurs="1" maxOccurs="1" />
    <xs:element name="Lng" type="xs:double" minOccurs="1" maxOccurs="1" />
    <xs:element name="Vrt" type="xs:double" minOccurs="1" maxOccurs="1" />
    <xs:element name="Pitch" type="xs:double" minOccurs="1" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ArmPositionsType">
  <xs:choice>
    <xs:element name="SpecialPosition" type="SpecialPositionType" minOccurs="0" maxOccurs="1" />
    <xs:element name="Positions" type="ArmAxes" minOccurs="0" maxOccurs="1" />
  </xs:choice>
</xs:complexType>

<xs:complexType name="KvBladePositionsType">
  <xs:annotation>
    <xs:documentation>
      Specify either blade tracking or blade positions.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:choice minOccurs="1" maxOccurs="1">
      <xs:element name="Tracking" type="xs:boolean" />
      <xs:element name="Positions" type="BladePositionsType" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ArmAxes">
  <xs:annotation>
    <xs:documentation>
      Axes definitions for a working position of an arm.
```

```
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!--units are deg or cm-->
    <xs:element minOccurs="1" maxOccurs="1" name="Lat" type="xs:double" />
    <xs:element minOccurs="1" maxOccurs="1" name="Lng" type="xs:double" />
    <xs:element minOccurs="1" maxOccurs="1" name="Vrt" type="xs:double" />
    <xs:element minOccurs="1" maxOccurs="1" name="Pitch" type="xs:double" />
  </xs:sequence>
</xs:complexType>

<!--
*****************************************************************************
***
***  Acquisition Parameters
***
*****************************************************************************
-->

<xs:complexType name="ImageMode">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Overwrite" type="ModeOverwrite" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="AcquisitionMode">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Overwrite" type="ModeOverwrite" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="ModeOverwrite">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="parameter" type="xs:string" use="required" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="Movie">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="Destination" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="Parameters" type="MovieParameters" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MovieParameters">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="Width" type="xs:int" />
    <xs:element minOccurs="0" maxOccurs="1" name="Height" type="xs:int" />
    <xs:element minOccurs="0" maxOccurs="1" name="FrameRate" type="xs:double" />
    <xs:element minOccurs="0" maxOccurs="1" name="Crop" type="xs:boolean" />
    <xs:element minOccurs="0" maxOccurs="1" name="Invert" type="xs:boolean" />
    <xs:element minOccurs="0" maxOccurs="1" name="PixelValue0" type="xs:int" />
    <xs:element minOccurs="0" maxOccurs="1" name="PixelValue255" type="xs:int" />
    <xs:element minOccurs="0" maxOccurs="1" name="CutOff" type="xs:int" />
    <xs:element minOccurs="0" maxOccurs="1" name="Weight" type="xs:double" />
  </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="HistogramRoi">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="X1" type="xs:int" />
    <xs:element minOccurs="1" maxOccurs="1" name="Y1" type="xs:int" />
    <xs:element minOccurs="1" maxOccurs="1" name="X2" type="xs:int" />
    <xs:element minOccurs="1" maxOccurs="1" name="Y2" type="xs:int" />
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="EFocalSpot">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Small"/>
    <xs:enumeration value="Large"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="EFluoroLevelControl">
  <xs:restriction base="xs:string">
    <xs:enumeration value="None"/>
    <xs:enumeration value="Low"/>
    <xs:enumeration value="High"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="ImageProcessing">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="Timeout" type="xs:unsignedInt" default="0"/>
    <xs:element minOccurs="1" maxOccurs="1" name="ModeId" type="xs:string" />
    <xs:element minOccurs="1" maxOccurs="1" name="AutoBeamOff" type="xs:boolean" />
    <xs:element minOccurs="0" maxOccurs="1" name="ConfidenceThreshold" type="xs:double" />
    <xs:element minOccurs="1" maxOccurs="1" name="Markers" type="MarkerDefinitions" />
  </xs:sequence>
</xs:complexType>


<xs:complexType name="MarkerDefinitions">
  <xs:choice minOccurs="1" maxOccurs="unbounded">
    <xs:element name="MarkerToleranceRadius" type="MarkerDefinitionToleranceRadius"/>
    <xs:element name="MarkerToleranceVolume" type="MarkerDefinitionToleranceVolume" />
  </xs:choice>
</xs:complexType>

<xs:complexType name="MarkerDefinition">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="Id" type="xs:string" />
    <xs:element minOccurs="1" maxOccurs="1" name="X" type="xs:double" />
    <xs:element minOccurs="1" maxOccurs="1" name="Y" type="xs:double" />
    <xs:element minOccurs="1" maxOccurs="1" name="Z" type="xs:double" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MarkerDefinitionToleranceRadius">
  <xs:complexContent>
    <xs:extension base="MarkerDefinition">
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="ToleranceRadius" type="xs:double" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
  <xs:complexType name="MarkerDefinitionToleranceVolume">
    <xs:complexContent>
      <xs:extension base="MarkerDefinition">
        <xs:sequence>
          <xs:element minOccurs="1" maxOccurs="1" name="Vertices" type="xs:string" />
          <xs:element minOccurs="1" maxOccurs="1" name="Indices" type="xs:string" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="MVParameters">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="Energy" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="DoseRate" type="xs:double" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="KVParameters">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="KiloVolts" type="xs:double" />
      <xs:element minOccurs="1" maxOccurs="1" name="MilliAmperes" type="xs:double" />
      <xs:element minOccurs="1" maxOccurs="1" name="MilliSeconds" type="xs:double" />
      <xs:element minOccurs="0" maxOccurs="1" name="FocalSpot" type="EFocalSpot" default="Large"/>
      <xs:element minOccurs="0" maxOccurs="1" name="FluoroLevelControl" type="EFluoroLevelControl"
default="Low"/>
      <xs:element minOccurs="0" maxOccurs="1" name="AutoBrightnessControl" type="xs:boolean" default="false"
/>
      <xs:element minOccurs="0" maxOccurs="1" name="DoseRecording" type="xs:boolean" default="true" />
      <xs:element minOccurs="0" maxOccurs="1" name="FrameRate" type="xs:int" default="0"/>
    </xs:sequence>
  </xs:complexType>


  <xs:complexType name="AcquisitionParameters">
    <xs:annotation>
      <xs:documentation>
      ImageMode       : Name of the image mode
      CalibrationSet  : Name of the calibration set used for image processing.
                        The default calibration set is called "DefaultCalibrationSetId"
      ImageDestination: The image destination defines the target destination
                        where XI should sent the acquired images. If it is not
                        defined the live destination is used. The system provides
                        predefined image destinations:
                        - AllImages           : Every frame is sent to XI Service
                                                and stored into the gallery.
                        - LiveImageDestination: Only the latest frmaes are sent to
                                                XI Service and only every few second
                                                one of these frame is stored into the
                                                gallery.
      NotificationDestination:
                        - NotificationDestination : Tracking data are sent to this
                                                    notification destination
      Movie:
                        - MovieDestination    : This is the default destination
                                                for movie encoding
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="ImageMode" type="ImageMode" />
```

```
    <xs:element minOccurs="0" maxOccurs="1" name="AcquisitionMode" type="AcquisitionMode" />
    <xs:element minOccurs="0" maxOccurs="1" name="CalibrationSet" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="unbounded" name="ImageDestination" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="unbounded" name="NotificationDestination" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="Movie" type="Movie" />
    <xs:element minOccurs="0" maxOccurs="1" name="RaiseFault" type="xs:boolean" />
    <xs:element minOccurs="0" maxOccurs="1" name="Cache" type="xs:boolean" />
    <xs:element minOccurs="0" maxOccurs="1" name="HistogramRoi" type="HistogramRoi" />
    <xs:element minOccurs="0" maxOccurs="1" name="ImageProcessing" type="ImageProcessing" />
    <xs:choice minOccurs="0" maxOccurs="1">
      <xs:element name="MV" type="MVParameters" />
      <xs:element name="KV" type="KVParameters" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<!--
*******************************************************************************
***
***  Motion Management Parameters
***
*******************************************************************************
-->

<xs:complexType name="Vector">
  <xs:annotation>
    <xs:documentation>
      Definition of a 3D vector matrix. Used by Placement to define
      the origin and orientation.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="X" type="xs:double" />
    <xs:element minOccurs="1" maxOccurs="1" name="Y" type="xs:double" />
    <xs:element minOccurs="1" maxOccurs="1" name="Z" type="xs:double" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Placement">
  <xs:annotation>
    <xs:documentation>
      A placement describes the translation (Origin) and orientation (AxisX,Y,Z).
      The placement itself does not define in which coordinate system the
      value are. The documentation of the use-case should include more
      information about the coordinate system.
      The three axis vectors must be orthogonal to each other and their
      length is one (norm vectors).
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="Origin" type="Vector" />
    <xs:element minOccurs="0" maxOccurs="1" name="AxisX" type="Vector" />
    <xs:element minOccurs="0" maxOccurs="1" name="AxisY" type="Vector" />
    <xs:element minOccurs="0" maxOccurs="1" name="AxisZ" type="Vector" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Coefficients">
  <xs:annotation>
    <xs:documentation>
      Coefficients for the amplitude fit model. For polynomial fits, a0... are used.
```

```
      For planar fit, the equation is a0 + a1*amplitude + a2*rate.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="a0" type="xs:double" />
    <xs:element minOccurs="0" maxOccurs="1" name="a1" type="xs:double" />
    <xs:element minOccurs="0" maxOccurs="1" name="a2" type="xs:double" />
    <xs:element minOccurs="0" maxOccurs="1" name="a3" type="xs:double" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="FitData">
  <xs:annotation>
    <xs:documentation>
      For the AmplitudeFit surrogate model. Supported fit types are:
      Static
      Linear
      Quadratic
      Cubic
      Planar
      See coefficients for a description of the used fitting equations.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="FitType" type="xs:string" />
    <xs:element minOccurs="1" maxOccurs="1" name="Coefficients" type="Coefficients" />
  </xs:sequence>
</xs:complexType>

<!-- Model system -->

<xs:complexType name="ModelSystem">
  <xs:annotation>
    <xs:documentation>
     The model system defines the relation between the fixed room system
     and the model system. The model system is always pinned to the couch.
     This means if the target moves with the couch the target position in
     the model system does not change.
     The couch position (lateral, longitudinal, vertical, rotation, pitch and roll)
     are in cm. If the couch position is not defined system is pinned to:
        - The defined couch position of the first control point
        - Or if there is not definition in the first control point
          it pins to the current couch position while loading this research beam
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="ModelSystem_pinned" type="Placement" />
    <xs:element name="CouchLat" minOccurs="0" maxOccurs="1" type="xs:double" />
    <xs:element name="CouchLng" minOccurs="0" maxOccurs="1" type="xs:double" />
    <xs:element name="CouchVrt" minOccurs="0" maxOccurs="1" type="xs:double" />
    <xs:element name="CouchRtn" minOccurs="0" maxOccurs="1" type="xs:double" />
    <xs:element name="CouchPit" minOccurs="0" maxOccurs="1" type="xs:double" />
    <xs:element name="CouchRol" minOccurs="0" maxOccurs="1" type="xs:double" />
  </xs:sequence>
</xs:complexType>

<!-- Surrogate models -->

<xs:complexType name="BasicSurrogateModel">
  <xs:annotation>
    <xs:documentation>
```

```
      The basic surrogate model defines a fix relation between the surrogate
      and the target. The relation TargetPosition_surrogate is defined in
      the fixed room system.
      By default the target position is equal to the surrogate
      position (identity for TargetPosition_surrogate).
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="TargetPosition_surrogate" type="Placement" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="AmplitudeFitSurrogateModel">
    <xs:annotation>
      <xs:documentation>
        The amplitude fit model maps the amplitude (and rate) of the motion model onto an internal axis.
        For each axis, a different fit can be given, see FitData comment for the equation.
        The surrogate model is executed in the couch-pinned model coordinate system, as defined by model
system.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="X" type="FitData" />
      <xs:element minOccurs="1" maxOccurs="1" name="Y" type="FitData" />
      <xs:element minOccurs="1" maxOccurs="1" name="Z" type="FitData" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="SurrogateModelPlugIn">
    <xs:annotation>
      <xs:documentation>
      Surrogate model plugin is an engineering only option
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
    </xs:sequence>
    <xs:attribute name="module" type="xs:string" use="required" />
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>

  <xs:complexType name="SurrogateModel">
    <xs:annotation>
      <xs:documentation>
      The surrogate model defines the relation between the surrogate and the target.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:choice minOccurs="1" maxOccurs="1">
        <xs:element name="Basic" type="BasicSurrogateModel" />
        <xs:element name="AmplitudeFit" type="AmplitudeFitSurrogateModel" />
        <xs:element name="PlugIn" type="SurrogateModelPlugIn" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <!-- Motion models -->

  <xs:complexType name="BasicMotionModel">
    <xs:annotation>
      <xs:documentation>
```

```
    The basic motion model does not detect any amplitude, phase or quality
    of the breathing pattern.
    </xs:documentation>
  </xs:annotation>
</xs:complexType>

<xs:complexType name="GatingMotionModel">
  <xs:annotation>
    <xs:documentation>
    The gating motion model calculated the amplitude, phase and quality of
    the breathing pattern by using SmartBreath. Furthermore the last max inhale
    and exhale, the running average of max exhale, running average of
    inhale and exhale period and the number of detected inhale peaks are
    reported.
    By default the amplitude direction is the Z axis in the model system.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="AmplitudeDirection_model" type="Vector" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MotionModelPlugIn">
  <xs:annotation>
    <xs:documentation>
    Motion model plugin is an engineering only option
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
  </xs:sequence>
  <xs:attribute name="module" type="xs:string" use="required" />
  <xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="MotionModel">
  <xs:annotation>
    <xs:documentation>
    The selected motion model defines how the current breathing motion
    is detected. See the description of each specific motion model for
    further information.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:choice minOccurs="1" maxOccurs="1">
      <xs:element name="Basic" type="BasicMotionModel" />
      <xs:element name="Gating" type="GatingMotionModel" />
      <xs:element name="PlugIn" type="MotionModelPlugIn" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<!-- Action windows -->

<xs:complexType name="AcquisitionTrigger">
  <xs:annotation>
    <xs:documentation>
    TriggerDelay   : Delay in sec before a trigger is released.
    TriggerOnEnter : Send a trigger on enter window
    TriggerOnExit  : Send a trigger on exit window
    SingleTrigger  : If true a trigger is only sent once for the whole
```

```
                    acquisition. If false a trigger is sent every time if the
                    window is enter or exit (depending on TriggerOnEnter
                    and TriggerOnExit).
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="TriggerDelay" type="xs:double" default="0" />
    <xs:element minOccurs="0" maxOccurs="1" name="TriggerOnEnter" type="xs:boolean" default="true" />
    <xs:element minOccurs="0" maxOccurs="1" name="TriggerOnExit" type="xs:boolean" default="false" />
    <xs:element minOccurs="0" maxOccurs="1" name="SingleTrigger" type="xs:boolean" default="true" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="AcquisitionTriggers">
  <xs:annotation>
    <xs:documentation>
    Use KV to sent the trigger to the kV image source and MV to trigger
    the MV image source.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:choice minOccurs="1" maxOccurs="unbounded">
      <xs:element name="MV" type="AcquisitionTrigger" />
      <xs:element name="KV" type="AcquisitionTrigger" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ActionWindow">
  <xs:annotation>
    <xs:documentation>
    Each defined action window (AW) is assigned to one axis. The AW defined
    the system behavior if the current axis value is within or outside
    of the defined border. It could hold the kV or MV beam (MVBeamImpact/KVBeamImpact).
    Additionally the action window could trigger an MV or kV image on the
    gate transition (enter or exit window).

    There are two different sets of axis ID's:
    - Global axes   : The global axes refers to a machine state
                      and are mainly used for MU, gantry or time
                      triggered imaging.
    - Tracking axes : The tracking axes are related to a actual value
                      determined by a running tracking acquisition.

    The supported global axis ID's are:
     - MU          : Used for delta triggered imaging based on delivered MUs
     - GantryAngle : Used for delta triggered imaging based on gantry
                     rotation [degree]
     - Time        : Used for delta triggered imaging based on treatment time.
                     The treatment time does not stop while the treatment
                     beam is held. The unit for the treatment time is seconds.
     - ConformityIndexOverArea    :
     - ConformityIndexUnderArea   :

    The supported tracking axis ID's are:
     - Amplitude       : Motion amplitude of model [cm]
     - Phase           : Motion phase of model [degree, 0-360]
     - Quality         : Quality of actual motion (in model system)
     - Target_fixed.X  : Target motion on X axis in fix system [cm]
     - Target_fixed.Y  : Target motion on Y axis in fix system [cm]
     - Target_fixed.Z  : Target motion on Z axis in fix system [cm]
```

```
       - Target_fixed.R  : Target radial motion in fix system [cm]
       - Target_fixed.AngleX : Target motion around X axis in fix system [degree]
       - Target_fixed.AngleY : Target motion around Y axis in fix system [degree]
       - Target_fixed.AngleZ : Target motion around Z axis in fix system [degree]
       - Target_model.X  : Target motion on X axis in model system [cm]
       - Target_ model.Y : Target motion on Y axis in model system [cm]
       - Target_ model.Z : Target motion on Z axis in model system [cm]
       - Target_ model.R : Target radial motion in model system [cm]
       - Target_ model.AngleX : Target motion around X axis in model system [degree]
       - Target_ model.AngleY : Target motion around Y axis in model system [degree]
       - Target_ model.AngleZ : Target motion around Z axis in model system [degree]

      Parameter description:
      Attribut axis : Name of the axis
      MVBeamImpact  : Hold the MV beam if tracking source is outside of the
                      action window.
      KVBeamImpact  : Hold the kV beam if tracking source is outside of the
                      action window.
      MotionCompensationImpact:
      LowerLimit    : Lower limit of the action window. Unit according to axis ID.
      UpperLimit    : Upper limit of the action window. Unit according to axis ID.
      Delta         : Delta value used for gantry, MU or time based triggered imaging
      EntryDelay    : Entry delay in milli seconds
      LingerTimeout : Linger timeout in milli seconds
      FaultOnExit   : Raise a fault on exit the action window
      AcquisitionTriggers: MV or kV acquisition triggers

      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="MVBeamImpact" type="xs:boolean" default="true" />
      <xs:element minOccurs="0" maxOccurs="1" name="KVBeamImpact" type="xs:boolean" default="false" />
      <xs:element minOccurs="0" maxOccurs="1" name="MotionCompensationImpact" type="xs:boolean"
default="false" />
      <xs:element minOccurs="1" maxOccurs="1" name="LowerLimit" type="xs:double" />
      <xs:element minOccurs="1" maxOccurs="1" name="UpperLimit" type="xs:double" />
      <xs:element minOccurs="0" maxOccurs="1" name="Delta" type="xs:double" />
      <xs:element minOccurs="0" maxOccurs="1" name="EntryDelay" type="xs:double" default="0" />
      <xs:element minOccurs="0" maxOccurs="1" name="LingerTimeout" type="xs:double" default="0" />
      <xs:element minOccurs="0" maxOccurs="1" name="FaultOnExit" type="xs:boolean" default="false" />
      <xs:element minOccurs="0" maxOccurs="1" name="AcquisitionTriggers" type="AcquisitionTriggers" />
    </xs:sequence>
    <xs:attribute name="axis" type="xs:string" use="required" />
  </xs:complexType>

  <xs:complexType name="ActionWindows">
    <xs:annotation>
      <xs:documentation>
      Basic Action Window:
      The basic action window is used to define a set of gating window.
      The beam is released automatically if the axis is within the gate again.

      Segmental Action Window:
      The beam is hold if the axis is outside the defined gate but not
      automatically resumed if the axis is inside the gate again. The user
      needs to give the OK before resuming.
      TODO: Verify the descriprion text for segmental
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:choice minOccurs="1" maxOccurs="unbounded">
```

```
      <xs:element name="Basic" type="ActionWindow" />
      <xs:element name="Segmental" type="ActionWindow" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<!-- Motion compensations -->

<xs:complexType name="BasicMotionCompensation">
  <xs:annotation>
    <xs:documentation>
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="TrackingSource" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1" name="CompensateRotation" type="xs:boolean" default="false" />
    <xs:element minOccurs="0" maxOccurs="2" name="Restriction_model" type="Vector" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MotionCompensationPlugIn">
  <xs:annotation>
    <xs:documentation>
    Motion compensation plugin is an engineering only option
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
  </xs:sequence>
  <xs:attribute name="module" type="xs:string" use="required" />
  <xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="MotionCompensation">
  <xs:annotation>
    <xs:documentation>
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:choice minOccurs="1" maxOccurs="1">
      <xs:element name="Basic" type="BasicMotionCompensation" />
      <xs:element name="PlugIn" type="MotionCompensationPlugIn" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<!-- Tracking sources -->

<xs:complexType name="TrackingSource">
  <xs:annotation>
    <xs:documentation>
      Attribute id          : Name of the tracking source. E.g. NDI, RTX1 or RTX2
      AcquisitionParameters : Defines the acquisition specific settings
                              as image mode name.
      SurrogateModel        : Model definition of the surrogate.
      MotionModel           : Model definition of the motion.
      TrackingActionWindows : Action windows which are related
                              to this tracking source (see description of ActionWindow)
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
```

```
      <xs:element minOccurs="1" maxOccurs="1" name="AcquisitionParameters" type="AcquisitionParameters" />
      <xs:element minOccurs="0" maxOccurs="1" name="SurrogateModel" type="SurrogateModel" />
      <xs:element minOccurs="0" maxOccurs="1" name="MotionModel" type="MotionModel" />
      <xs:element minOccurs="0" maxOccurs="1" name="TrackingActionWindows" type="ActionWindows" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
  </xs:complexType>

  <!-- Motion management parameters -->

  <xs:complexType name="MotionManagementParameters">
    <xs:annotation>
      <xs:documentation>
        Define XI specific motion model parameters

        ModelSystem         : Defines the model system
        TrackingSource      : Setup parameters for involved tracking sources
        MotionCompensation  : Defines the motion compensation model
        GlobalActionWindows : Action windows which are not related
                              to a tracking source (see description of ActionWindow)
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="ModelSystem" type="ModelSystem" />
      <xs:element minOccurs="0" maxOccurs="unbounded" name="TrackingSource" type="TrackingSource" />
      <xs:element minOccurs="0" maxOccurs="1" name="MotionCompensation" type="MotionCompensation" />
      <xs:element minOccurs="0" maxOccurs="1" name="GlobalActionWindows" type="ActionWindows" />
    </xs:sequence>
  </xs:complexType>

  <!--
  ********************************************************************************
  ***
  ***  iTool specific parameters
  ***
  ********************************************************************************
  -->

  <xs:complexType name="iTools">
    <xs:sequence>
      <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
    </xs:sequence>
  </xs:complexType>

  <!--
  ********************************************************************************
  ***
  ***  Imaging Points and Parameters
  ***
  ********************************************************************************
  -->

  <xs:complexType name="ImagingPoints">
    <xs:annotation>
      <xs:documentation>
        Sequence of imaging points.
        An initial imaging point at Cp=0 is mandatory for all imaging axes, which are specified in a
subsequent imaging point.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
```

```
      <xs:element minOccurs="1" maxOccurs="unbounded" name="ImagingPoint" type="ImagingPoint" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ImagingPoint">
    <xs:annotation>
      <xs:documentation>
        Imaging point.
        Refers to fractional control point indices, e.g. 3.5 means with all Clinac axes averaged between
control points with indices 3 and 4.
        In case of a beam group (or superbeam), Cp refers to the fractional control point index of the
combined beam.
        Example:
        Beam group with 3 beams.
        First beam has 5 control points:        0 &#8804; Cp &#8804; 4.
        Second beam has 2 control points:       5 &#8804; Cp &#8804; 6.
        Third beam has 3 control points:        7 &#8804; Cp &#8804; 9.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!--units are deg or cm, varian internal format -->
      <xs:element minOccurs="1" maxOccurs="1" name="Cp" type="xs:double" />
      <xs:element minOccurs="0" maxOccurs="8" name="Acquisition" type="Acquisition" />
      <xs:element minOccurs="0" maxOccurs="8" name="AcquisitionStart" type="Acquisition" />
      <xs:element minOccurs="0" maxOccurs="8" name="AcquisitionStop" type="Acquisition" />
      <!-- kV filter settings -->
      <xs:element minOccurs="0" maxOccurs="1" name="KvFilters" type="KvFiltersPositionType" />
      <!--desired arm /kv blade positions before the image acquisition -->
      <xs:element minOccurs="0" maxOccurs="1" name="KvBlades" type="KvBladePositionsType" />
      <xs:element minOccurs="0" maxOccurs="1" name="Mvd" type="ArmPositionsType" />
      <xs:element minOccurs="0" maxOccurs="1" name="Kvd" type="ArmPositionsType" />
      <xs:element minOccurs="0" maxOccurs="1" name="Kvs" type="ArmPositionsType" />
      <!--desired arm position after the image acquisition -->
      <xs:element minOccurs="0" maxOccurs="1" name="MvdAfter" type="ArmPositionsType" />
      <xs:element minOccurs="0" maxOccurs="1" name="KvdAfter" type="ArmPositionsType" />
      <xs:element minOccurs="0" maxOccurs="1" name="KvsAfter" type="ArmPositionsType" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Acquisition">
    <xs:annotation>
      <xs:documentation>
        Acquisition parameters for one image source.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="AcquisitionId" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="AcquisitionSpecs" type="AcquisitionSpecs" />
      <xs:element minOccurs="0" maxOccurs="1" name="AcquisitionParameters" type="AcquisitionParameters" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="AcquisitionSpecs">
    <xs:annotation>
      <xs:documentation>
        TODO: Verify with Daniel Wong is description is correct

        The acquisition specs defines the coordination between XI and SPV.
        The meaning of the handshake flag is different for kV and MV imaging.

        Handshake for MV imaging:
```

```
        This is used for radshot imaging where SPV places the required MV
        dose and afterwards triggers XI to readout the detector. For continuous
        or Dosimetry imaging this flag has to be false.

        Handshake for kV imaging:
        If the handshake flag is true the SPV stops all axis until XI has
        finished the kV acquisition (if the imaging point uses &lt;Acquisition&gt;)
        or the acquisition is started/stopped (if the imaging point uses
        &lt;AcquisitionStart&gt; or &lt;AcquisitionStop&gt;).

        KV:         True if image with kV Beam.
        MVDose:     Estimated total amount of MV dose required in order to acquire the image.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="Handshake" type="xs:boolean" />
      <xs:element minOccurs="0" maxOccurs="1" name="KV" type="xs:boolean" />
      <xs:element minOccurs="0" maxOccurs="1" name="MVDose" type="xs:double" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="DuringTreatment">
    <xs:annotation>
      <xs:documentation>
        Type for imaging during treatment.

        Images may be acquired during treatment.
        If the beam is paused while XI is in an acquiring state, the current image(s) will be dropped and
        the treatment beam can be resumed under SPV control at the control point where it was paused.

        If the system is equipped for imaging, but no images have to be acquired for a treatment beam,
        this type is used at well. In this case, imaging points shall contain the initial arm positions,
        but no acquisitions.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence></xs:sequence>
  </xs:complexType>

  <xs:complexType name="OutsideTreatment">
    <xs:annotation>
      <xs:documentation>
        Type for imaging outside treatment.

        Supervisor shall terminate the beam as soon as one of the following condition is fulfilled:
        1. All acquisitions are completed (normal termination).
        2. The maximum MU limit (MaxMu) is reached.
        3. The operator stops the beam.
        4. A fault occurs.

        Supervisor beam state never changes to 'paused'.
        The reason of termination can be determined from the supervisor beam history.

        Note:
        The corresponding beam has the following characteristics.
        1. There may be no MV energy defined (e.g. kV imaging or dark field acquisition), MaxMu shall be 0 in
this case.
        2. The meterset values in the control points have no semantics.
        3. There may be 2 successive control points with no changes in any value.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
```

```
      <xs:element minOccurs="1" maxOccurs="1" name="MaxMu" type="xs:double" />
    </xs:sequence>
  </xs:complexType>

  <xs:element name="ImagingParameters">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>
          Imaging parameters associated with a (super-)beam.
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:choice minOccurs="1" maxOccurs="1">
          <xs:element name="DuringTreatment" type="DuringTreatment" />
          <xs:element name="OutsideTreatment" type="OutsideTreatment" />
        </xs:choice>
        <xs:element minOccurs="0" maxOccurs="1" name="LatchBEL" type="xs:boolean" default="true"/>
        <xs:element minOccurs="0" maxOccurs="1" name="LatchKVBEL" type="xs:boolean" default="true"/>
        <xs:element minOccurs="1" maxOccurs="1" name="ImagingPoints" type="ImagingPoints" />
        <xs:element minOccurs="1" maxOccurs="1" name="ImagingTolerances" type="ImagingTolerances" />
        <xs:element minOccurs="0" maxOccurs="1" name="MotionManagementParameters"
type="MotionManagementParameters" />
        <xs:element minOccurs="0" maxOccurs="1" name="iTools" type="iTools" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>


  <!--
  ********************************************************************************
  ***
  ***  Control Points
  ***
  ********************************************************************************
  -->

  <xs:element name="A" type="xs:string">
    <xs:annotation>
      <xs:documentation>
        Contains an array of MLC positions delimited by spaces for Bank A.
        The first leaf in the array is A-1, and the last leaf is A-40, A-60,
        or A-26.      The positions are defined in the array as double, in
        1 cm units, in Varian scale.
        Values represent logical (as opposed to physical) positions in the Isocenter Plane.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="B" type="xs:string">
    <xs:annotation>
      <xs:documentation>
        Contains an array of MLC positions delimited by spaces for Bank B.
        See Documentation for "A"
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:complexType name="MlcPositionsType">
    <xs:annotation>
      <xs:documentation>
        Type for a set of MLC Positions
```

```
      ID = 1 means primary MLC
      ID = 2 mean secondary MLC
      In a given Control Point, either all MLC Leaf Positions are present or "Mlc" element is missing.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="ID" type="xs:int" minOccurs="1" maxOccurs="1" />
    <xs:sequence>
      <xs:element ref="B" minOccurs="1" maxOccurs="1" />
      <xs:element ref="A" minOccurs="1" maxOccurs="1" />
    </xs:sequence>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SubBeamType">
  <xs:annotation>
    <xs:documentation>
      Indicates the start of a new "Sub" Beam

      Seq - sequence number of the "Sub"  Beam within the Group (0,1,2,...)
      Name = name of the "Sub"    Beam within the Group
      MaxRadTime = maximum radiation time for this Sub beam (in seconds).
      The controller also calulates the nominal radiation time.
      The smaller of the two values is used.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="Seq" type="xs:unsignedInt" minOccurs="1" maxOccurs="1" />
    <xs:element name="SubbeamGUID" type="xs:string" minOccurs="0" maxOccurs="1" />
    <xs:element name="Name" type="xs:string" minOccurs="0" maxOccurs="1" />
    <xs:element name="MaxRadTime" type="xs:double" minOccurs="0" maxOccurs="1" />
    <xs:element name="TrackingTrainingOnly" type="xs:boolean" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:element name="Cp">
  <xs:annotation>
    <xs:documentation>
      A single ControlPoint

      General Control Point Rules:
      1) Unspecified (i.e. unplanned) STATIC Axes,
      should NOT APPEAR in ANY Control Point.    The Control System will not enforce
      any tolerance restrictions on such axes.  However, te Control System still will ensure that
      these  axes do not move from wherever they are placed once it goes to READY.
      NOTE: MLC may be unspecified STATIC by this rule.
      2) Specified STATIC Axes, whose initial positions are contained in the Beam need only
      appear in the First Control Point. The controller will ensure that they
      do not move from wherever they are placed once it goes to READY.
      3) Specified DYNAMIC Axes, whose position "program" is contained in the Beam need only
      appear in the First Control Point and any subsequent Control Point where
      there is a change in their position.  The controller will ensure that they
      follow their position "program" during the treatment).
      4) Only one independent jaw of a pair may be appear in a Control Point.
      It is not necessary for both to appear. This could occur if only one jaw is dynamic.
      5) MU is cumulative. From one control point to the next, it can only go up
      or stay the same.    It cannot decrease.


      Units:
      1) Energy  - A unique signature for the energy of the form "dds" where
```

```
      dd = 0-99,
      and s = 'x', 'e', or 'h', where
      x -- MV X-Rays
      e -- MeV electrons
      h -- Mev HDTSe- electons.
      k -- KV beams
      Examples: "6x" (6 MV X-Rays) and "12e" (12 Mev electrons) and "0k" (kv beam)
      2) Mu - 1 MU
      3) Axis Positions - 1 deg and 1 cm in Varian Internal Scale
      4) Dose Rate - 1 MU/min (max dose rate)

      First Control Point Rules:
      1) Must contain "Energy".
      2) Must contain "Mu" = 0.
      3) Must contain "DRate".
      4) Must contain "SubBeam"

      Subsequent Control Points:
      1) Must contain "Energy" only if it is changing
      2) Must contain "Mu" only if it is changing (i.e. not a "no dose" segment).
      3) Must contain "DRate" only if it is changing
      4) Must contain axes whose positions are changing (dynamic axes).
      5) May not contain axes (including MLC) which were not contained in the first Control Point.

      Name Attribute:
      The Name Attribute is the name of an Individual Beam with a Group of Beams.
      The presence of a name attribute indicates the start of a new Individual Beam.
      If there is only one Beam in the Group, only the 1st Cp will have the Name Attribute.

      TreatProgressEvent: Any Control Point which is marked with a "TreatProgressEvent" element causes
      the Control System to broadcast a "TreatProgress" event.  (See schema for "TreatProgress" event
      for more details.)
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="TreatProgressEvent" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="SubBeam" type="SubBeamType" minOccurs="0" maxOccurs="1" />
      <xs:element name="Energy" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="Mu" type="unsignedDouble" minOccurs="0" maxOccurs="1" />
      <xs:element name="DRate" type="unsignedDouble" minOccurs="0" maxOccurs="1" />
      <xs:element name="GantryRtn" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CollRtn" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchVrt" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchLat" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchLng" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchRtn" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchPit" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchRol" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="Y1" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="Y2" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="X1" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="X2" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="Mlc" type="MlcPositionsType" minOccurs="0" maxOccurs="2" />
      <xs:element name="Phase" type="doublePhase" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="ControlPoints">
  <xs:annotation>
```

```
    <xs:documentation>
      Segment Treatment Table
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:annotation>
      <xs:documentation>
        Cp - A single control point
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element ref="Cp" minOccurs="2" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>


<!--*****************************************************************************
            ***
            ***  Beam Tracking Definitions
            ***
            *****************************************************************************
            -->
<xs:complexType name="ConformityType">
  <xs:annotation>
    <xs:documentation>
      Define the parameters of the real-time treatment delivery confirmity measure.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="OverExposure" type="unsignedDouble" minOccurs="1" maxOccurs="1" />
    <xs:element name="UnderExposure" type="unsignedDouble" minOccurs="1" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="TrackingMotionType">
  <xs:annotation>
    <xs:documentation>
      If the axis should follow the tracking target or the baseline position.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction   base="xs:string">
    <xs:enumeration value="target"/>
    <xs:enumeration value="baseline"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="TrackingAxis">
  <xs:annotation>
    <xs:documentation>
      Define the axis that is allowed to track.
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="Tol" type="unsignedDouble"/>
  <xs:attribute name="MotionType" type="TrackingMotionType"/>
</xs:complexType>

<xs:complexType name="TrackingMLC">
  <xs:annotation>
    <xs:documentation>
      If this is in the beam, tracking is enabled for the collimation system.
```

```
        Specifying an ExpectedTargetSpeed > 0 will result in circulating leaf pairs in reserve.
        YTargetRange is an additional paramter for the leaf pairs in reserve which bounds
        circulation up to the given range in Y direction.
      </xs:documentation>
    </xs:annotation>
    <xs:attribute name="MotionType" type="TrackingMotionType"/>
    <xs:attribute name="ID" type="xs:int" fixed="1"/>
    <xs:attribute name="OpenUpCarriages" type="unsignedDouble"/>
    <xs:attribute name="ExpectedTargetSpeed" type="unsignedDouble"/>
    <xs:attribute name="YTargetRange" type="unsignedDouble"/>
  </xs:complexType>

  <xs:complexType name="TrackingPhase">
    <xs:annotation>
      <xs:documentation>
        If in the beam, phase tracking is enabled. Every control point has to contain valid phase
information.
        Phase is given in degrees [0.0;360.0[.
        Trajectory execution speed will be adapted to match the plan phase with the online measured phase
from XI.
        MaxPhaseLag is the maximum lag in phase space that is tolerated; if the plan execution lags more
behind,
        the trajectory engine will hold and wait for the next breathing phase.
      </xs:documentation>
    </xs:annotation>
    <xs:attribute name="MaxPhaseLag" type="doublePhase"/>
  </xs:complexType>

  <xs:complexType name="TrackingAxisList">
    <xs:annotation>
      <xs:documentation>
        Define the motion axes that are allowed to track.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="CouchVrt" type="TrackingAxis" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchLat" type="TrackingAxis" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchLng" type="TrackingAxis" minOccurs="0" maxOccurs="1" />
      <xs:element name="Y12" type="TrackingAxis" minOccurs="0" maxOccurs="1" />
      <xs:element name="X12" type="TrackingAxis" minOccurs="0" maxOccurs="1" />
      <xs:element name="Mlc" type="TrackingMLC" minOccurs="0" maxOccurs="1" />
      <xs:element name="Phase" type="TrackingPhase" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>

  <!--
  ********************************************************************************
  ***
  ***  Research Beam Top Level and Root
  ***
  ********************************************************************************
  -->

  <!-- Tracking -->

  <xs:element name="Tracking">
    <xs:annotation>
      <xs:documentation>
        Define the tracking enable status plus the tracking capability ratio.

        The following rules are enforced:
```

```
      - couch axes require either the axis Tol or the ConformityTol to be set
      - jaw tracking (X12 and Y12) requires MLC tracking (Mlc)
      - Mlc tracking requires ConformityTol to be set
      - any tracking axis must have at least initial positions in the first control point
      - if ConformityTol is specified there must be at least initial positions of Mlc and jaws.

      Tracking capability ratio range between 0 and 1.0, with 1.0 means tracking trajectory would
      be generated with full capabilities of axes.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Axes" type="TrackingAxisList" minOccurs="0" maxOccurs="1" />
      <xs:element name="ConformityTol" type="ConformityType" minOccurs="0" maxOccurs="1" />
      <xs:element name="InitialCapabilityRatio" type="doubleRatio" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>


<!-- MLC Model -->

<xs:simpleType name="MLCModelType">
  <xs:annotation>
    <xs:documentation>
      Define the model of MLC that is required for the plan. Possible MLC model Enumeration
      May need to moved to CommonDefinitions.xsd.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction   base="xs:string">
    <xs:enumeration value="NDS80"/>
    <xs:enumeration value="NDS120"/>
    <xs:enumeration value="NDS120HD"/>
  </xs:restriction>
</xs:simpleType>

<!-- Tolerance Table -->

<xs:element name="TolTable">
  <xs:annotation>
    <xs:documentation>
      Tolerance Table

      RULES:
      1. If an individual axis has a specified zero tolerance value or a missing tolerance element,
      the Control System will impose its tight internal tolerance on that axis.
      2. If the entire tolerance table is missing, the Control System will impose its tight
      internal tolerance on all axes.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <!--units are deg or cm ... -->
      <xs:element name="GantryRtn" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CollRtn" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchVrt" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchLat" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchLng" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchRtn" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchPit" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchRol" type="xs:double" minOccurs="0" maxOccurs="1" />
```

```
      <xs:element name="Y12" type="xs:double" minOccurs="0" maxOccurs="1" />
      <!-- ... independent Y tolerance -->
      <xs:element name="X12" type="xs:double" minOccurs="0" maxOccurs="1" />
      <!-- ... independent X tolerance -->
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Maximal Velocity Table -->

<xs:element name="VelTable">
  <xs:annotation>
    <xs:documentation>
      Max Velocity Table:
      This table can be used to specify top velocities of axes to be used in the treatment.

      RULES:
      1. If an individual axis has a specified max velocity value, then the Control
      System will use the smaller of the specified max velocity and the maximum supported
      max velocity of that axis.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <!--units are 1 deg/sec or 1 cm/sec ... -->
      <xs:element name="GantryRtn" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CollRtn" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchVrt" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchLat" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchLng" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchRtn" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchPit" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="CouchRol" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="X1" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="X2" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="Y1" type="xs:double" minOccurs="0" maxOccurs="1" />
      <xs:element name="Y2" type="xs:double" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Accessories -->

<xs:element name="Accs">
  <xs:annotation>
    <xs:documentation>
      The planned Accessory Codes for each slot.

      RULES:
      1. A missing element for a slot means that ANY accessory can be installed in the slot
      without causing an interlock; i.e. "don't care".
      2. The value '0' (zero) in a slot means that no accessory must be installed in the slot.
      3. Any other integer value in a slot means that an accessory with that accessory code must
      be installed in that slot.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Acc1" type="xs:int" minOccurs="0" maxOccurs="1" />
      <xs:element name="Acc2" type="xs:int" minOccurs="0" maxOccurs="1" />
      <xs:element name="Acc3" type="xs:int" minOccurs="0" maxOccurs="1" />
```

```
      <xs:element name="Acc4" type="xs:int" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Beam Hold Devices -->

<xs:element name="Dev">
  <xs:annotation>
    <xs:documentation>
      The planned mode flags for the particular beam hold device.  This
      schema should be able to accommodate EXGI, which is an external
      gating device, or any other beam hold device and define its usage
      connected directly to the BGM.
      MVBeamImpactFlag  : This beam hold device does hold the treatment beam if true.
      MotionImpactFlag  : This beam hold device does stop motion if true.

      Table of possible combinations :
      ¦ MV Beam Impact ¦ Motion Impact
      -------------------------------------------------
      Outside Treatment ¦ False          ¦ False
      ¦ True           ¦ False
      ¦ False          ¦ True
      ¦ True           ¦ True
      -----------------------------------------------------------
      During Treatment ¦ False          ¦ False
      ¦ True           ¦ True
      | True           | False

      If MVBeamImapct or MotionImpact are not present, then they default to true.

      SyncStop is an experimental features, which ramps down the plan execution including the beam
      before the beam hold state is reached. Use with MVBeamImpact=false (since BGM delays
      beam-hold until told by SPV), and MotionImpact=true.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="Id" type="xs:int"/>
    <xs:attribute name="MVBeamImpact" type="xs:boolean" use="optional"/>
    <xs:attribute name="MotionImpact" type="xs:boolean" use="optional"/>
    <xs:attribute name="SyncStop" type="xs:boolean" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="BeamHoldDevices">
  <xs:annotation>
    <xs:documentation>
      The planned beam hold devices.

      RULES:
      1. A missing beam hold device means that the beam hold devie must not be ready.
      2. A present beam hold device must be ready
      3. The MV beam gets held if the beam hold device requests to held the beam.
      4. Id attribute specifies the id of the beam hold device (number between 0 and 15)
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Dev" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
```

```
  </xs:element>

  <!-- Root -->

  <xs:element name="SetBeam">
    <xs:annotation>
      <xs:documentation>
        The root element of the research beam

        Id - unique Beam ID
        TolTable - the tolerance table for the beam
        ControlPoints - the set of control points for the beam
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SchemaVersion" type="xs:string" minOccurs="0" maxOccurs="1" />
        <xs:element name="Id" type="xs:int" minOccurs="1" maxOccurs="1" />
        <xs:element name="GUID" type="xs:string" minOccurs="0" maxOccurs="1" />
        <xs:element name="TrajectoryUploadInfo" type="xs:anyURI" minOccurs="0" maxOccurs="1" />
        <xs:element name="TreatmentMode" type="TreatmentModeType" minOccurs="0" maxOccurs="1" />
        <xs:element name="MLCModel" type="MLCModelType" minOccurs="1" maxOccurs="1" />
        <xs:element ref="TolTable" minOccurs="0" maxOccurs="1" />
        <xs:element ref="VelTable" minOccurs="0" maxOccurs="1" />
        <xs:element ref="Accs" minOccurs="1" maxOccurs="1" />
        <xs:element ref="ControlPoints" minOccurs="1" maxOccurs="1" />
        <xs:element ref="ImagingParameters" minOccurs="0" maxOccurs="1" />
        <xs:element ref="BeamHoldDevices" minOccurs="0" maxOccurs="1" />
        <xs:element ref="Tracking" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

APPENDIX B: Accessories List

Accessory	Code (Range)	Slot	Applicator Type
NoAccy1	0	InterfaceMount	
NoAccy2	0	AccessoryMount	
NoAccy3	0	ElectronAperture	
NoAccy4	0	Compensator Mount	
UnkAcc1	4095	InterfaceMount	
UnkAcc2	4095	AccessoryMount	
UnkAcc3	4095	ElectronAperture	
UnkAcc4	4095	Compensator Mount	
UnkAcc5	127	ElectronAperture	
W15R30U	3920	InterfaceMount	
W15L30U	3921	InterfaceMount	
W15IN30U	3922	InterfaceMount	
W15OUT30U	3923	InterfaceMount	

W30R30U	3924	InterfaceMount	
W30L30U	3925	InterfaceMount	
W30IN30U	3926	InterfaceMount	
W30OUT30U	3927	InterfaceMount	
W45R20U	3912	InterfaceMount	
W45L20U	3913	InterfaceMount	
W45IN20U	3914	InterfaceMount	
W45OUT20U	3915	InterfaceMount	
W60R15U	3900	InterfaceMount	
W60L15U	3901	InterfaceMount	
W60IN15U	3902	InterfaceMount	
W60OUT15U	3903	InterfaceMount	
W15R30L	3920	Compensator Mount	
W15L30L	3921	Compensator Mount	
W15IN30L	3922	Compensator Mount	
W15OUT30L	3923	Compensator Mount	
W30R30L	3924	Compensator Mount	
W30L30L	3925	Compensator Mount	
W30IN30L	3926	Compensator Mount	
W30OUT30L	3927	Compensator Mount	
W45R20L	3912	Compensator Mount	
W45L20L	3913	Compensator Mount	
W45IN20L	3914	Compensator Mount	
W45OUT20L	3915	Compensator Mount	
W60R15L	3900	Compensator Mount	
W60L15L	3901	Compensator Mount	
W60IN15L	3902	Compensator Mount	
W60OUT15L	3903	Compensator Mount	
A06	4000	AccessoryMount	ELECTRON_SQUARE
A10	4001	AccessoryMount	ELECTRON_SQUARE
A15	4002	AccessoryMount	ELECTRON_SQUARE
A20	4003	AccessoryMount	ELECTRON_SQUARE
A25	4004	AccessoryMount	ELECTRON_SQUARE
A10X6	4006	AccessoryMount	ELECTRON_RECT
FT.POINTER	8167	InterfaceMount	
FFDA(A10+)	4094	ElectronAperture	
FFDA(A06)	126	ElectronAperture	
e-ARC Applicator	4094	AccessoryMount	
PF Grat Upper	3801	InterfaceMount	
PF Grat Lower	3792	Compensator Mount	
CustomBlockTray	1 - 1664	AccessoryMount	

UpperCustomComp	1 - 364	InterfaceMount	
LowerCustomComp	1 - 364	Compensator Mount	
CustomFFDA6	1 - 125	ElectronAperture	
CustomFFDA	3328 - 3707	ElectronAperture	
IsoCal Plate	3600	InterfaceMount	
CustomSRSAppl	3265 - 3326	AccessoryMount	STEREOTACTIC
CustomSRSMount	3264	InterfaceMount	