# Description of XML used in CASPR

## 1 Introduction

In CASPR, XML scripting is used for a number of applications including describing models and providing information for the GUI settings. In this document, a description of the different XML files used and their expected format is provided.

## 2 Model Description using XML

CASPR makes use of XML scripts in order to specify CDPR models. A new CDPR model can be added to CASPR by creating the following XML files:

1. **bodies.xml**: The rigid body structure of the CDPR, such as the inertia properties and the type of joints. This file also describes the operational space if it is defined for the new model.
2. **cables.xml**: The set of cable arrangements for the cable attachment locations and cable properties.
3. **trajectories.xml**: The set of possible trajectories that the robot may execute.

The sections below provide details on the meaning of xml tags and structures for each of these documents.

### 2.1 Bodies XML

Code Sample 1 shows an example bodies XML file. The following can be noted about the structure of this file

1. The document is enclosed by the tag `bodies_system`.
2. Within the `bodies_system` tag there is a tag `links` which contains link specification information.
3. Operational space sets are also specified within the `bodies_system` tag using the `operational_spaces` tag. Different operational spaces can be specified for the same robot.

Listing 1: Example Bodies XML File.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE bodies_system SYSTEM "../../../templates/bodies.dtd">
<bodies_system>
    <links display_range="-3.0 3.0 -3.0 3.0 -3.0 3.0" view_angle="0 0">
      <link_rigid num="1" name="R_Y 1">
        <joint type="R_Y" q_initial="0" q_min="-3.1416" q_max="3.1416"/>
        <physical>
          <mass>1</mass>
          <com_location>0.0 0.0 0.5</com_location>
          <end_location>0.0 0.0 1.0</end_location>
          <inertia ref="com">
            <Ixx>0.083333</Ixx>
            <Iyy>0.0</Iyy>
            <Izz>0.083333</Izz>
            <Ixy>0.0</Ixy>
            <Ixz>0.0</Ixz>
            <Iyz>0.0</Iyz>
          </inertia>
        </physical>
        <parent>
          <num>0</num>
          <location>0.0 0.0 0.0</location>
        </parent>
      </link_rigid>
      <link_rigid num="2" name="R_Y 2">
        <joint type="R_Y" q_initial="0" q_min="-3.1416" q_max="3.1416"/>
```

```
        <physical>
          <mass>1</mass>
          <com_location>0.0 0.0 0.5</com_location>
          <end_location>0.0 0.0 1.0</end_location>
          <inertia ref="com">
            <Ixx>0.083333</Ixx>
            <Iyy>0.0</Iyy>
            <Izz>0.083333</Izz>
            <Ixy>0.0</Ixy>
            <Ixz>0.0</Ixz>
            <Iyz>0.0</Iyz>
          </inertia>
        </physical>
        <parent>
          <num>1</num>
          <location>0.0 0.0 1.0</location>
        </parent>
      </link_rigid>
    </links>
    <operational_spaces default_operational_set="test">
        <operational_set id="test">
            <position marker_id="1" name="test1">
                <link>2</link>
                <offset>0.0 0.0 1.0</offset>
                <axes active_axes="x"/>
            </position>
        </operational_set>
    </operational_spaces>
</bodies_system>
```

A more detailed list of each of the elements used by the bodies and the requirements on those elements is provided below:

- `bodies_system` - The root node for the file.
  *Required Child Elements:* `links`.
  *Optional Child Elements:* `operational_spaces`.

- `links` - This elements acts as a container for the set of all links that describe the mechanism.
  *Required Child Elements:* At least one `link_rigid` element.
  *Required Attributes:*

  - `display_range` - The range for plotting specified in the format $[\underline{x}, \overline{x}, \underline{y}, \overline{y}, \underline{z}, \overline{z}]$ where $x$, $y$ and $z$ represent the associated axes of the base frame.
  - `view_angle` - The viewing angle specified in the format $[\theta, \phi]$ where $\theta$ is the azimuthal angle and $\phi$ is the elevation.

- `link_rigid` - The physical parameters and interconnection information for a single rigid link.
  *Required Child Elements:* `joint`, `physical` and `parent`.
  *Required Attributes:*

  - `num` - A numerical identifier for the link.
  - `name` - A text identifier for the link.

- `joint` - The joint that the rigid link is connected to.
  *Required Attributes:*

  - `type` - The joint type given as an enum which matches those given in `src/Model/Bodies/Joints/JointType.m`.
  - `q_initial` - A vector of the initial pose of the joint (this vector must contain the same number of elements as the defined number of variables for the joint).
  - `q_min` - A vector of the minimum allowable pose of the joint (this vector must contain the same number of elements as the defined number of variables for the joint).
  - `q_max` - A vector of the maximum allowable pose of the joint (this vector must contain the same number of elements as the defined number of variables for the joint).

- `physical` - A container for the physical parameter information associated with a link.
  *Required Child Elements:* `mass`, `com_location`, `end_location` and `inertia`.

- `mass` - The mass of the link (in kg).

- `com_location` - The displacement vector ${}^{i}\mathbf{r}_{P_iG_i}$ (in m). This corresponds to the position of the centre of mass relative to the joint location expressed in the reference frame of the link.
- `end_location` The displacement vector ${}^{i}\mathbf{r}_{P_iE_i}$ (in m). This corresponds to the position of the link endpoint relative to the joint location expressed in the reference frame of the link.
- `inertia` - The inertia of the mechanism. The components of the inertia matrix are expressed as follows

$$I_{\texttt{ref}} = \begin{bmatrix} Ixx & Ixy & Ixz \\ Ixy & Iyy & Iyz \\ Ixz & Iyz & Izz \end{bmatrix}. \tag{1}$$

  *Required Child Elements:* `Ixx`, `Ixy`, `Ixz`, `Iyy`, `Iyz` and `Izz`.
  *Required Attributes:*
  - `ref` - The reference for the inertia tensor given as "com" or "joint" as required.
- `parent` - A container element to contain the information that describes the parent link information.
  *Required Child Elements:* `num` and `location`.
- `num` - The num tag indicates the parent link by providing the numerical identifier for that link.
- `location` - The location tag contains the vector ${}^{p(i)}\mathbf{r}_{P_{p(i)}P_i}$. That is the position of the joint relative to the frame of reference of the parent link $p(i)$.
- `operational_spaces` - A container for all operational space sets.
  *Required Child Elements:* At least one `operational_set`.
  *Required Attributes:*
  - `default_operational_set` - The default operational space set. To be used unless otherwise specified.
- `operational_set` - A set of operational space points.
  *Required Child Elements:* At least one marker object. Supported types currently consist of `position`, `orientation_euler_xyz` and `pose_euler_xyz`.
  *Required Attributes:*
  - `id` - A text identifier for this operational space set.
- `position` - A translational operational space marker.
  *Required Child Elements:* `link`, `offset` and `axes`.
  *Required Attributes:*
  - `marker_id` - A numerical identifier for the marker.
  - `name` - A text identifier for the marker
- `orientation_euler_xyz` - A rotational operational space marker.
  *Required Child Elements:* `link` and `axes`.
  *Required Attributes:*
  - `marker_id` - A numerical identifier for the marker.
  - `name` - A text identifier for the marker
- `pose_euler_xyz` - A 6 DoF operational space marker.
  *Required Child Elements:* `link`, `offset`, `axes`.
  *Required Attributes:*
  - `marker_id` - A numerical identifier for the marker.
  - `name` - A text identifier for the marker
- `link` - The link that the operational space marker is connected to.
- `offset` - The vector ${}^{e(i)}\mathbf{r}_{P_{e(i)}E_i}$. That is the position of the marker relative to the frame of reference of the attached link $e(i)$.
- `axes` - The axes to consider for the operational space marker.
  *Required Attributes:*
  - `active_axes` - The currently active axes. 'x' for the ${}^{0}x_1$ axis, 'y' for the ${}^{0}x_2$ axis, 'z' for the ${}^{0}x_3$ axis, 'a' for rotation about ${}^{0}x_1$, 'b' for rotation about ${}^{0}x_2$ and 'g' for rotation about ${}^{0}x_3$.

## 2.2 Cables XML

Code Sample 2 shows an example cables XML file. The following can be noted about the structure of this file

1. The document is enclosed by the tag `cables`.
2. Within the `cables` tag there is a tag associated with each set of cables (`cable_set`).
3. Within the `cable_set` tag there is a tag associated with each cable.
4. The single cable is itself further defined by a two tags `properties` which contains information associated with the physical properties of the cable and `attachments` which contains information about the attachments of the cable.

Listing 2: Example Cables XML File.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cables SYSTEM "../../../templates/cables.dtd">

<cables default_cable_set="basic">
  <cable_set id="basic">
    <cable_ideal name="cable 1" attachment_reference="com">
      <properties>
        <force_min>0.1</force_min>
        <force_max>1000</force_max>
      </properties>
      <attachments>
        <attachment>
          <link>0</link>
          <location>0.0 0.0 0.0</location>
        </attachment>
        <attachment>
          <link>1</link>
          <location>-0.125 0 0</location>
        </attachment>
      </attachments>
    </cable_ideal>
    <cable_ideal name="cable 2" attachment_reference="com">
      <properties>
        <force_min>0.1</force_min>
        <force_max>1000</force_max>
      </properties>
      <attachments>
        <attachment>
          <link>0</link>
          <location>1.0 0.0 0.0</location>
        </attachment>
        <attachment>
          <link>1</link>
          <location>0.125 0 0</location>
        </attachment>
      </attachments>
    </cable_ideal>
    <cable_ideal name="cable 3" attachment_reference="com">
      <properties>
        <force_min>0.1</force_min>
        <force_max>1000</force_max>
      </properties>
      <attachments>
        <attachment>
          <link>0</link>
          <location>1.0 1.0 0.0</location>
        </attachment>
        <attachment>
          <link>1</link>
          <location>0.125 0 0</location>
        </attachment>
      </attachments>
    </cable_ideal>
    <cable_ideal name="cable 4" attachment_reference="com">
      <properties>
        <force_min>0.1</force_min>
        <force_max>1000</force_max>
      </properties>
      <attachments>
```

```
        <attachment>
          <link>0</link>
          <location>0.0 1.0 0.0</location>
        </attachment>
        <attachment>
          <link>1</link>
          <location>-0.125 0 0</location>
        </attachment>
      </attachments>
    </cable_ideal>
  </cable_set>
</cables>
```

A more detailed glossary of these tags in addition to all other tags used by the file is provided below.

- `cables` - This is the root node element.
  *Required Child Elements:* At least one `cable_set`.
  *Required Attributes:*
  - `default_cable_set` - The default cable set to use in the case in which no other set is specified.
- `cable_set` - A container for a set of cables.
  *Required Child Elements:* At least one cable. Currently supported cable types include `cable_ideal`, `cable_linear_spring`, `cable_passive_linear_spring`, `cable_vsd_torsion_spring` and `cable_vsd_flexure_linear`.
  - `id` - The text identifier for the cable set.
- `cable_ideal`, `cable_linear_spring`, `cable_passive_linear_spring`, `cable_vsd_torsion_spring` and `cable_vsd_flexure_linear` - A cable of the defined type.
  *Required Child Elements:* `properties` and `attachments`.
  *Required Attributes:*
  - `name` - A text identifier for the cable.
  - `attachment_ref` - The attachment reference point for the cable. Set to be either 'joint' or 'com'.
- `properties` - The physical property information for the cable.
  *Required Child Elements:*
  - `force_min` and `force_max` (in N) for `cable_ideal`.
  - `force_min` and `force_max` (in N) and `K` (in N/m) for `cable_linear_spring`.
  - `l0` (in m) and `K_cable` (in N/m) for `cable_passive_linear_spring`.
  - `force_min` and `force_max` (in N), `K_cable` (in N/m) and `vsd_force_deformation_relation` for `cable_vsd_flexure_linear`.
  - `force_min` and `force_max` (in N), `K_cable` (in N/m), `num_torsion_springs`, `torsion_spring_stiffness` (in N/rad) and `torsion_spring_length` for `cable_vsd_torsion_spring`.
- `attachments` - A container for the attachment information for a cable. *Required Child Elements:* At least two `attachment` and/or `base_rotating_pulley` tags.
- `attachment` - A tag that contains an individual attachment location (`link` and `location`).
- `link` - The link that the attachment connects to (0 for the base).
- `location` The vector ${}^{i}\mathbf{r}_{P_i A_{ijk}}$ which is the location of the attachment in the frame of reference of the link.

## 2.3 Trajectories XML

Code Sample 3 shows an example trajectories XML file. The following can be noted about the structure of this file

1. The document is enclosed by the tag `trajectories`.
2. The `trajectories` tag contains a `joint_trajectories` tag and optionally a `operational_trajectories`
3. Within these tags there is a tag associated with each trajectory (`trajectory`) which contains an attribute of the `time_step`.

4. The trajectory tag is then described by a `points` tag containing a set of `point` tags.

Listing 3: Example Trajectory XML File.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE trajectories SYSTEM "../../../templates/trajectories.dtd">
<trajectories>
    <joint_trajectories>
        <quintic_spline_trajectory id="traj_test" time_definition="absolute" time_step="
            0.00667">
            <points>
                <point>
                    <q>0.0 0.0 0.0 0.0</q>
                    <q_dot>0.0 0.0 0.0 0.0</q_dot>
                    <q_ddot>0.0 0.0 0.0 0.0</q_ddot>
                </point>
                <point time="4">
                    <q>-0.17453 0.0 0.0 0.17453</q>
                    <q_dot>0.0 0.0 0.0 0.0</q_dot>
                    <q_ddot>0.0 0.0 0.0 0.0</q_ddot>
                </point>
            </points>
        </quintic_spline_trajectory>
        <linear_spline_trajectory id="traj_test_Sl" time_definition="absolute" time_step="
            0.00667">
            <points>
                <point>
                    <q>0.0 0.0 0.0 0.0</q>
                </point>
                <point time="4">
                    <q>0.2 0.0 -0.2 0.0</q>
                </point>
                <point time="8">
                    <q>0.2 0.0 0.2 -0.0</q>
                </point>
                <point time="10">
                    <q>0.0 -0.0 0.2 -0.0</q>
                </point>
                <point time="14">
                    <q>-0.0 0.0 -0.2 0.0</q>
                </point>
                <point time="16">
                    <q>-0.2 0.0 -0.2 0.0</q>
                </point>
                <point time="20">
                    <q>-0.2 0.0 0.2 0.0</q>
                </point>
                <point time="24">
                    <q>-0.0 0.0 -0.0 0.0</q>
                </point>
            </points>
        </linear_spline_trajectory>
    </joint_trajectories>
    <operational_trajectories>
      <quintic_spline_trajectory id="simple_motion" time_step="0.005" time_definition="
          relative">
        <points>
          <point>
            <y>0.2 0.3 0.3</y>
            <y_dot>0.0 0.0 0.0</y_dot>
            <y_ddot>0.0 0.0 0.0</y_ddot>
          </point>
          <point time="5.0">
            <y>0.3 0.1 0.2</y>
            <y_dot>0.0 0.0 0.0</y_dot>
            <y_ddot>0.0 0.0 0.0</y_ddot>
          </point>
        </points>
      </quintic_spline_trajectory>
    </operational_trajectories>
</trajectories>
```

A more detailed glossary of these tags in addition to all other tags used by the file is provided below.

- `trajectories` - This is the root node element.
  *Required Child Elements:* `joint_trajectories`.

*Optional Child Elements:* `operational_trajectories`.

- `joint_trajectories` A container element for all joint space trajectories.
  *Required Child Elements:* At least on trajectory element. Currently supported trajectory elements include `quintic_spline_trajectory`, `cubic_spline_trajectory`, `linear_spline_trajectory`, `cubic_spline_average_velocity_trajectory` and `parabolic_blend_trajectory`.

- `quintic_spline_trajectory`, `cubic_spline_trajectory`, `linear_spline_trajectory`, `cubic_spline_average_velocity_trajectory` and `parabolic_blend_trajectory` - A trajectory in either joint or operational space.
  *Required Child Elements:* `points`.
  *Required Attributes:*
  - `id` - A text identifier for the trajectory.
  - `time_step` - The time step between each trajectory point.
  - `time_definition` - An indicator of how to read the time for each point. Given as "absolute" or "relative".
  - `blend_time_default` (only required in `parabolic_blend_trajectory`) - The default time for blends.

- `points` - A container for `point` tags.
  *Required Child Elements:* At least one `point`.

- `point` - The pose information a trajectory point. *Required Child Elements:*
  - `q`, `q_dot` and `q_ddot` or `y`, `y_dot` and `y_ddot` for `quintic_spline_trajectory`.
  - `q` and `q_dot` or `y` and `y_dot` for `cubic_spline_trajectory`.
  - `q` or `y` for `linear_spline_trajectory`, `cubic_spline_average_velocity_trajectory` and `parabolic_blend_trajectory`.

  *Required Attributes:*
  - `time` - The time associated with a given point. This is not required for the first point option (where the time is set to 0).
  - `blend_time` (only required in `parabolic_blend_trajectory`) - The time associated with the blending.

- `q`, `q_dot`, `q_ddot` - The joint space pose or its derivatives. One parameter should be given for each degree of freedom.

- `y`, `y_dot`, `y_ddot` - The operational space pose or its derivatives. One parameter should be given for each operational space degree of freedom.

# 3 GUI Specifications using XML

XML is also used in CASPR in order to provide the GUI with necessary information regarding which options to load for each box. Written below is a short description of the elements contained in the following XML files:

- Control XML,
- Dynamics XML,
- Kinematics XML,
- Workspace XML.

## 3.1 Control XML

- `simulator` - This is the root node element.
  *Required Child Elements:* At least one of each of `control_class`, `solver_class` and `plot_functions`.

- `control_class` - A controller to be used by the GUI. Note controllers should only be added to the GUI after testing in scripts.
  *Required Attributes:*

  - `id` - A text string identifying the controller. This should match the class name for the controller.

- `solver_class` - An inverse dynamics solver to be used by the GUI. Note inverse dynamics solvers should only be added to the GUI after testing in scripts.
  *Required Attributes:*

  - `id` - A text string identifying the inverse dynamics solver. This should match the class name for the inverse dynamics solver.

- `plot_functions` - A container element to contain all possible plotting functions.
  *Required Child Elements:* At least one `plot_function` object.

- `plot_function` - A function to be used for plotting. Note this should only be added to the GUI after testing in scripts.
  *Required Child Elements:* `figure_quantity`.
  *Required Attributes:*

  - `type` - A text string identifying the plotting function. This should correspond to a pre-existing plot function defined in a Simulator class.

- `figure_quantity` - The number of figures that are to be generated by the plotting function.

## 3.2 Dynamics XML

Since the control XML supports the use of inverse dynamics, it provides all the elements that are used in the dynamics XML. As such the inverse dynamics XML uses all elements mentioned within the control XML glossary with the exception of `control_class`.

## 3.3 Kinematics XML

- `simulator` - This is the root node element.
  *Required Child Elements:* At least one of each of `solver_class` and `plot_functions`.

- `solver_class` - A forward kinematics solver to be used by the GUI. Note forward kinematics solvers should only be added to the GUI after testing in scripts.
  *Required Attributes:*

  - `id` - A text string identifying the forward kinematics solver. This should match the class name for the forward kinematics solver.

- `plot_functions` - A container element to contain all possible plotting functions.
  *Required Child Elements:* At least one `plot_function` object.

- `plot_function` - A function to be used for plotting. Note this should only be added to the GUI after testing in scripts.
  *Required Child Elements:* `figure_quantity`.
  *Required Attributes:*

  - `type` - A text string identifying the plotting function. This should correspond to a pre-existing plot function defined in a Simulator class.

- `figure_quantity` - The number of figures that are to be generated by the plotting function.

## 3.4 Workspace XML

- `simulator` - This is the root node element.
  *Required Child Elements:* At least one of each of `workspace_condition`, `workspace_metrics`, `grid_types` and `plot_functions`.

- `workspace_condition` - A workspace condition to be used by the GUI. Note workspace conditions should only be added to the GUI after testing in scripts.
  *Required Attributes:*

  - `id` - A text string identifying the workspace condition. This should match the class name for the workspace condition.

- `workspace_metrics` - A container element to hold workspace metrics.
  *Required Child Elements:* At least one `workspace_metric`.

- `workspace_metric` - A workspace metric to be used by the GUI. Note workspace metrics should only be added to the GUI after testing in scripts.

- `grid_types` - A container element to hold grid types.
  *Required Child Elements:* At least one `grid_type`.

- `grid_type` - A grid type to be used by the GUI. Note grid types should only be added to the GUI after testing in scripts.

- `plot_functions` - A container element to contain all possible plotting functions.
  *Required Child Elements:* At least one `plot_function` object.

- `plot_function` - A function to be used for plotting. Note this should only be added to the GUI after testing in scripts.
  *Required Child Elements:* `figure_quantity`.
  *Required Attributes:*

  – `type` - A text string identifying the plotting function. This should correspond to a pre-existing plot function defined in a Simulator class.

- `figure_quantity` - The number of figures that are to be generated by the plotting function.