[User Guide](http://www.opendial-toolkit.net/user-manual)‎ > ‎

Dialogue domains

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| This section details how to practically encode dialogue domains for OpenDial using XML.  **1. General structure**  A dialogue domain in OpenDial follows the skeleton below:  <domain>    <initialstate>     <!--(optional) initial state variables -->   </initialstate>    <parameters>     <!--(optional) prior distributions for rule parameters -->   </parameters>    <model trigger="trigger variables for model 1">     <!--probabilistic rules for model 1 -->   </model>    <model trigger="trigger variables for model 2">     <!-- probabilistic rules for model 2 -->   </model>    ...     <model trigger="trigger variables for model n">     <!-- probabilistic rules for model n -->   </model>     <settings>     <!--(optional) domain-specific settings -->   </settings>  </domain>  The settings, initial state and parameters can be left out of the domain specification if empty. The number of rule-structured models is arbitrary.  For more complex domains, the domain specification can be split in several files through the import marker:      <import href="path to another file" />  Numerous examples of dialogue domains can be found in the directory domains and test/domains of the base directory.  **XML format for <domain>:**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | <initialstate> | Element | 0-1 | Initial state for the dialogue domain | | <parameters> | Element | 0-1 | Prior parameter distributions | | <import href="..."/> | Element | 0-n | Import of other XML files | | <model trigger="..."> | Element | 0-n | Dialogue model | | <settings> | Element | 0-1 | Domain-specific system settings |   **2. Initial state**  The initial state for the domain defines the variables included in the dialogue state upon starting the dialogue system. Each variable has a particular identifier and a probability distribution.  Variables with a discrete range of values are defined as categorical tables:  <variable id="variable\_id">   <value prob="probability for first value">first value</value>   <value prob="probability for second value">second value</value>   ...   <value prob="probability for the nth value">nth value</value> </variable>  Probability values must be comprised between 0 and 1. If the total probability amounts to less than 1, OpenDial automatically adds an empty value (None) for the remaining probability mass. If the prob attribute is omitted, the value is assumed to have a probability 1.  Here is a simple example of state variable:  <variable id="userIntention">   <value prob="0.5">Want(Object\_A)</value>   <value prob="0.3">Want(Object\_B)</value> </variable>  Probability distributions can also be defined for a continuous range, using the XML element <distrib type="..."> (see below).  **XML format for <initialstate>:**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | <variable id="..."> | Element | 0-n | State variable |   **XML format for <variable> in <initialstate>:**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | id | Attribute | 1 | Variable label | | <value prob="p"> | Element | 1-n | Possible value for the variable with probability p. If the attribute prob is omitted, the probability is assumed to be 1. | | ***or*** <distrib type="..."> | Element | 0-1 | *cf. below* |  |  | | --- | | **IMPORTANT NOTE**: Generally speaking, variable can have arbitrary identifiers, but a couple of special characters should be avoided. Variables should ***not*** include primes ('), curly brackets ({,}) or square brackets ([,]), as these are used internally in OpenDial. Furthermore, variables ending with ^p, ^t and ^o have a special function: ^p denotes [predictive variables](http://www.opendial-toolkit.net/user-manual/advanced-modelling-1#6._Predictive_variables), ^t denotes temporary variables that are deleted immediately after each update loop, and ^o denotes observation variables for [user simulators](http://www.opendial-toolkit.net/user-manual/parameter-estimation#With_simulated_dialogues).  Some variable values also have a special meaning in OpenDial: "None" denotes an "empty" value, and values between square brackets [ ] denote sets of elements. |     **3. Parameters**  Probabilistic rules can include parameters whose values is initially unknown and must be estimated from data. As OpenDial adopts a Bayesian learning approach, each parameter must be associated with an prior distribution over its (usually continuous) range of possible values.  **XML format for <parameters>:**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | <variable id="..."> | Element | 0-n | State variable |   Parameters are defined in exactly the same way as state variables. Their distributions are defined in a parametric manner:   * **Uniform distributions** are defined with two parameters min and max. The distribution *U(-1,3)* is thus encoded as:   <variable id="uniform\_example">   <distrib type="uniform">     <min>-1</min>     <max>3</max>   </distrib> </variable>   * **Gaussian distributions**[[1]](http://www.opendial-toolkit.net/user-manual/dialogue-domains#%5B1%5D) are defined with two parameters mean and variance -- for instance, *N(2,4)* is encoded as:   <variable id="gaussian\_example">   <distrib type="gaussian">     <mean>2</mean>     <variance>4</variance>   </distrib> </variable>   * **Dirichlet distributions**. A Dirichlet distribution is a multivariate continuous distribution. It is often employed to describe the prior parameter distribution of categorical/multinomial distributions. Dirichlet distributions are defined by a list of *alpha* values (one for each dimension). For instance, the 3-dimensional distribution *Dirichlet(1,1,2)* is expressed as:   <variable id="dirichlet\_example">  <distrib type="dirichlet">   <alpha>1</alpha>     <alpha>1</alpha>     <alpha>2</alpha>   </distrib> </variable>  **4. Models**  A dialogue model is essentially defined as a set of probabilistic rules combined with one or more "trigger variables" that defines when the rules are to be applied:  <model trigger="trigger variable(s)">   <rule id="rule 1">       ...  </rule>   <rule id="rule 2">       ...  </rule>   ...   <rule id="rule n">       ...  </rule>  </model>  The trigger variables must be separated by a comma. The rules can either encode probability or utility rules, as we explain below.  **XML format for <model>:**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | id | Attribute | 0-1 | (optional) name for the model | | trigger | Attribute | 1 | Comma-separated list of trigger variables | | <rule> | Element | 1-n | Probability or utility rule |   *Probability rules*  Probability rules express how a subset of state variables (the "input variables" of the rule) affect the probability distribution over some other state variables (the "output variables"). The output variables may either already exist in the dialogue state (in which case their content is erased) or represent new variables to include in the dialogue state.  Probability rules are structured as an **if...then...else** construction:  if (condition c1) then   P(effect e1) = ...     P(effect e2) = ...   ... else if (condition c2) then   ... else   ...  In XML, these probability rules are expressed as (ordered) list of cases. Each case has a (possibly empty) condition and a list of alternative effects (each with a particular probability).  Here is one concrete example of probability rule (corresponding to the rule r1 in [Lison (2014)](http://folk.uio.no/plison/pdfs/thesis/thesis-plison2014.pdf), p. 65):  <rule id="r1">   <case>     <condition>       <if var="Rain" value="false"/>       <if var="Weather" value="hot"/>     </condition>     <effect prob="0.03">       <set var="Fire" value="true"/>     </effect>     <effect prob="0.97">       <set var="Fire" value="false"/>     </effect>   </case>   <case>     <effect prob="0.01">       <set var="Fire" value="true"/>     </effect>     <effect prob="0.99">       <set var="Fire" value="false"/>     </effect>    </case> </rule>  Rule r1 simply indicates that the probability of a fire if there is no rain and a hot weather is 0.03, while this probability is 0.01 in other cases.  In some circumstances, one may want to enforce a particular dominance hierarchy among the rules (in order to ensure that some rules have priority over others if they are triggered simultaneously). This can be specified using the priority attribute, taking an integer value (where 1 indicates the highest priority).  **XML format for <rule>:**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | id | Attribute | 0-1 | (optional) name for the rule | | priority | Attribute | 0-1 | (optional) integer indicated the priority level of the rule (where 1 is highest) | | <case> | Element | 1-n | List of rule cases |   **XML format for <case>:**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | <condition> | Element | 0-1 | Condition for the case. If omitted, OpenDial assumes an empty (i.e. trivially true) condition. | | <effect> | Element | 1-n | List of alternative effects for the case |   We now detail how the conditions and effects are practically specified.  *Conditions*  As exemplified in the rule above, the condition XML node is composed of a list of basic conditions.  **XML format for <condition>:**[[2]](http://www.opendial-toolkit.net/user-manual/dialogue-domains#%5B2%5D)   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | operator | Attribute | 0-1 | (Optional) logical operator. Possible values are "and" and "or". Default value is "and". | | <if ...> | Element | 0-n | Basic condition. |   Each basic condition is written as an <if .../> markup with three basic attributes:  **XML format for <if .../>:**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | var | Attribute | 1 | Variable label | | relation | Attribute | 0-1 | (Optional) binary relation to satisfy. Default relation is equality. Admissible relations are:   * = (equality) * != (inequality) * &lt; (lower than) * &gt; (greater than) * contains (contains element or substring) * !contains (does not contain element or substring) * in (is contained in) * !in (is not contained in) | | value | Attribute | 1 | Variable value to check |   *Effects*  Each case contains one or more (alternative) effects. Each effect has a particular probability of occurrence. This probability can be specified by hand, as in the example above:      <effect prob="0.03">       <set var="Fire" value="true"/>     </effect>  When the effect does not specify any prob attribute, the effect is assumed to have a probability 1. When the total probability for all effects is lower than 1, an empty effect is implicitly assumed to cover the remaining probability mass.  The probability of a particular effect can also be a parameter. In this case, each case with *n*alternative effects is associated with a *nth* dimensional Dirichlet distribution that express the possible values for the effect probabilities. For instance, the effect probabilities in rule r1can be rewritten as:  <rule id="r1">   <case>     <condition>       <if var="Rain" value="false"/>       <if var="Weather" value="hot"/>     </condition>     <effect prob="firstdirichlet[0]">       <set var="Fire" value="true"/>     </effect>     <effect prob="firstdirichlet[1]">       <set var="Fire" value="false"/>     </effect>   </case>   <case>     <effect prob="seconddirichlet[0]">       <set var="Fire" value="true"/>     </effect>     <effect prob="seconddirichlet[1]">       <set var="Fire" value="false"/>     </effect>    </case> </rule>  Note the brackets after the parameter name to refer to a specific dimension of the multivariate Dirichlet.  **XML format for <effect> (for probability rules):**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | prob | Attribute | 0-1 | Probability for the effect (either fixed or parameter). Default value is 1. | | <set ...> | Element | 1-n | Basic effect |   Inside each effect is a list of basic assignment of values to variables. Each assignment is defined by a <set.../> markup with two attributes: var and value.  **XML format for <set .../> (for probability rules):**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | var | Attribute | 1 | Variable label | | value | Attribute | 1 | Variable value |   *Utility rules*  Rule can also be employed to express utility models. A utility rule defines the utility of particular actions (from the system perspective) depending on particular state variables. The general skeleton remains similar to probability rules, with the difference that effects are this time associated to particular utilities instead of probabilities. Here is an example of utility rule (rule r2 of [Lison (2014)](http://folk.uio.no/plison/pdfs/thesis/thesis-plison2014.pdf), p. 69):  <rule id="r2">   <case>     <condition>       <if var="Fire" value="true"/>     </condition>     <effect util="5">       <set var="Tanker" value="drop-water"/>     </effect>     <effect util="-5">       <set var="Tanker" value="wait"/>     </effect>   </case>   <case>     <effect util="-1">     <set var="Tanker" value="drop-water"/>     </effect>     <effect util="0">     <set var="Tanker" value="wait"/>     </effect>   </case> </rule>  Rule r2 indicates that the utility of the drop-water action is +5 is there is a fire (and -1 otherwise), and that the utility of wait is -5 is there is a fire and 0 otherwise.  Conditions are defined similarly to probability rules. Effects also have a similar structure, with one exception: the prob attribute is replaced by util. The variables specified in the effect (Tanker in the above example) are action variables.  As for probability rules, utilities can be fixed or correspond to parameters to estimate. For instance, rule r2 can include four parameters that denote the respective utility of the system actions depending on the situation:  <rule id="r2">   <case>     <condition>       <if var="Fire" value="true"/>     </condition>     <effect util="firstgaussian">       <set var="Tanker" value="drop-water"/>     </effect>     <effect util="secondgaussian">       <set var="Tanker" value="wait"/>     </effect>   </case>   <case>     <effect util="thirdgaussian">     <set var="Tanker" value="drop-water"/>     </effect>     <effect util="fourthgaussian">     <set var="Tanker" value="wait"/>     </effect>   </case> </rule>  **XML format for <effect> (for utility rules):**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | util | Attribute | 0-1 | Utility for the action (either fixed or parameter). Default value is 0. | | <set ...> | Element | 1-n | Basic effect |   **XML format for <set ... /> (for utility rules):**   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Cardinality** | **Description** | | var | Attribute | 1 | variable label (action variable) | | value | Attribute | 1 | Variable value |     **5. Settings**  In addition to an initial state, parameters and rule-structured models, a dialogue domain can also include particular system settings to override the default values.[[3]](http://www.opendial-toolkit.net/user-manual/dialogue-domains#%5B4%5D)  The settings are defined as simple list of elements:  <settings> <property1>value for property1</property1> <property2>value for property2</property2> .... </settings>  These properties can also be modified through the GUI or by adding a -Dproperty=value flag to the command line.  **XML format for <settings>:**  *(partial list, see*[*Settings.java*](https://github.com/plison/opendial/blob/master/src/opendial/Settings.java)*for all details)*   |  |  |  |  | | --- | --- | --- | --- | | **Content** | **XML Type** | **Value** | **Description** | | gui | Element | Boolean | Whether to start the GUI or not | | user | Element | String | Variable label for the user utterance | | system | Element | String | Variable label for the system utterance | | samples | Element | Integer | Number of samples to use when sampling | | timeout | Element | Integer | Maximum sampling time (in milliseconds) | | modules | Element | Comma-separated list | List of classes implementing Module to attach to the  system |   [1] Multivariate Gaussian distributions can also be defined. In this case, the scalar values for the mean and variance are replaced by vector values in the form <mean>[v1,v2,..,vn]</mean>. Multivariate Gaussian distributions support for the moment only distributions with a diagonal covariance (i.e. independent Gaussians).  [2] Conditions can also include the nested operators <and>, <not> and <or> (cf. [Advanced modelling: nested conditions](http://www.opendial-toolkit.net/user-manual/advanced-modelling-1#6._Nested_conditions)).  [3] The default settings can be found in the file resources/settings.xml. |