# XDI Policy

# Introduction

To meet the security and privacy requirements of XDI-based systems acting for different authorities, the XDI protocol enables systems to precisely describe access and usage rights to the data they control. In order for these rights to be enforced uniformly by the all XDI authorities to which they are granted, XDI authorization is described in XDI itself. This includes the ability to express any policy governing authorization and for policies to reference data, variables, relations, and other statements in the relevant XDI graphs. XDI’s primary policy building block is the link contract, a mechanism that can work equally well in either a centralized, or distributed (peer to peer) model.

# Related Publications

The XDI Bindings Specification enables both direct communications between native XDI endpoints and the conveyance of XDI messages through people’s browsers.

The XDI Message Specification enables access operations, which are evaluated against policies. Policy establishment also requires message flows.

The XDI Cryptographic Syntaxes Specification can be used to add integrity and auditable to XDI Policy operations.

# Terminology

**XDI policy:** The rules and practices by which XDI authorities protect information graphs from unauthorized access operations, or unauthorized usage.

**Operational policies:** Control access to the graph. They can be specified in XDI link contracts or XDI messages and optionally make use of conditional policy expressions.

**Usage policies:** Specify a requesting authority’s (RA’s) permissions to use data from the authorizing authority’s (AA’s) graph once the data is accessed. In many cases the obligation to honor a usage policy must be specified in a legal (or other) agreement for it to be effective.

**Conditional policy expressions:** Boolean reserved words and patterns for use in XDI policies.

**Other policy variables:** Time and other variables may be used in policy expressions.

**XDI message:** An XDI subgraph exchanged between XDI authorities to accomplish an XDI operation ($get, $get, etc.) against information in a target graph.

**Link contract:** An XDI subgraph used for access control, or authorization between an AA and an RA. It is a machine-readable agreement describing data shared and permissions granted by one XDI authority to another. A link contract describes both the policy that must be satisfied by an XDI message to be accepted and the permissions granted to the RA if those conditions are met. Link contracts may be used to apply and enforce any type of policy over shared data and messages, including security, privacy, re-sharing, synchronization, and termination.

**Link contract establishment:** The process of setting up a link contract between an AA and RA using additional link contract templates, governors and instance patterns.

# Policy Model

By default, no one has access to data in an XDI graph unless access is granted through a link contract. An XDI authority SHOULD create a root link contract to express its access rights to its whole graph. If parts of the graph are to be shared publicly with external services, the controlling authority SHOULD create public link contract (allowing either anonymous or anonymous but authenticated) access to them. If parts of the graph are to be shared with other authorities, the controlling authority SHOULD create generic or specific link contract instances for them through link contract establishment exchanges.

Thus, link contracts may be set at various contexts of a graph. The closest link contract to a sub-graph controls its access or usage policies and overrides all higher link contracts except for the root link contract.

All access to the graph other than by its controlling authority’s software occurs via XDI messages. XDI messages may be submitted to the graph for policy evaluation through local APIs or over a network interface.

An XDI graph may be owned by one authority, such as an enterprise or service provider, yet contain sub-graphs that are owned by other authorities. These sub-graphs are expressed as inner roots to mark them clearly as belonging to the separate authority. They contain one or more link contract instances allowing data to be retrieved from or copied from the authoritative source.

# Policy Evaluation

**XDI operational policies** are expressed in XDI link contracts or XDI messages and may be evaluated against an evaluation context. The result is always a boolean value (true or false). Two elements are needed to set the evaluation context:

1. The XDI graph that is the target of an XDI message (called the **target graph**).
2. The XDI message that is meant to be executed against the target graph (called the **input message**).

An XDI link contract policy is part of the target graph. It provides a way for an XDI endpoint to determine if the input message is authorized. An XDI message policyis part of the input message. It provides a way for an XDI client to request execution of the input message only if certain conditions are met. XDI link contract policies and XDI message policies are evaluated by the XDI endpoint that hosts the target graph and receives the input message.

An **XDI data usage policy** controls usage of the requested data outside the scope of XDI enforcement. Data usage policies are attached to a link contract controlling access to part of the target graph. XDI data usage policies are evaluated by the authority that holds the data to which the XDI data usage policy applies. This may be the XDI endpoint that hosts the target graph, or it may be any other XDI endpoint or XDI client which has retrieved the data (and the XDI data usage policy along with it).

Note: More work on data usage policies may come in the future, including the possibility of attaching them to the data itself rather than the link contract. Usage policies are supported in this working draft primarily as a basic privacy mechanism to prevent undesired copying of data when RAs and AAs are part of a trust framework supporting these policies.

# Policy Expression Requirements

Policy expressions MUST follow the patterns and include applicable reserved words as specified below.

## Policy Expression Pattern Requirements

A policy MUST consist of one or more XDI statements in the following pattern:

**Policy Pattern**

<--policy-context--> $if<--boolean-context-->/<--operator-->/<--condition-->

Where:

* <--policy-context--> is the context of the policy expression, e.g. an XDI link contract or an XDI message or a data usage context
* <--boolean-context--> is an optional hierarchy of subcontexts expressing the XDI boolean operators (**$and**, **$or**, **$not**) needed to produce a boolean logic tree for policy evaluation (see below).
* <--operator--> is either a boolean operator (**$true**, **$false**), or an XDI operation being requested by a message (e.g., **$get**, **$set**, **$del**, **$do<--extension-->**).
* <--condition--> is either a comparison condition (**$is**, **$equals**, **matches**, **$greater**, **$lesser**), or an arbitrary XDI statement.

## Conditional Operations Policy Expression Dictionary Requirements

The following reserved words are proposed for policy expression.

### $if

* **$if** defines the root context node of a policy expression subgraph.
* **$if** MUST be a child context of a **$do** context (the root context of an XDI link contract or the operation context of an XDI message) or a $use context (the data usage context)
* **$if** MUST be an entity singleton.
* **$if** MAY have zero or more boolean subcontexts.
* **$if** MAY have zero or more operator predicates.
* **$if** evaluates to true, if
  + ANY of its boolean subcontexts evaluate to true, OR
  + ANY of its operator predicates evaluate to true.
* **$if** evaluates to false, if it has neither boolean subcontexts nor operator predicates.

### <--boolean-context-->

#### **$and**

* **$and** is the boolean context expressing logical conjunction.
* **$and** MAY be an entity singleton for a single logical conjunction.
* **$and** MUST be an entity collection if there are multiple logical conjunctions that must be evaluated independently.
* **$and** MAY have zero or more boolean subcontexts.
* **$and** MAY have zero or more operator predicates.
* **$and** evaluates to true, if
  + ALL of its boolean subcontexts evaluate to true, AND
  + ALL of its operator predicates evaluate to true
* **$and** evaluates to true, if it has neither boolean subcontexts nor operator predicates.

#### **$or**

* **$or** is the boolean context defining logical disjunction.
* **$or** MAY be an entity singleton for a single logical disjunction.
* **$or** MUST be an entity collection if there are multiple logical disjunctions that must be evaluated independently.
* **$or** MAY have zero or more boolean subcontexts.
* **$or** MAY have zero or more operator predicates.
* **$or** evaluates to true, if
  + ANY of its boolean subcontexts evaluate to true, OR
  + ANY of its operator predicates evaluate to true.
* **$or** evaluates to false, if it has neither boolean subcontexts nor operator predicates.

#### **$not**

* **$not** is the boolean context defining logical negation.
* **$not** MAY be an entity singleton for a single logical negation.
* **$not** MUST be an entity collection if there are multiple logical negations that must be evaluated independently.
* **$not** MUST either have exactly one boolean subcontext, OR have exactly one operator predicate.
* **$not** evaluates to true, if
  + its single boolean subcontext evaluates to false, OR
  + its single operator predicate evaluates to false.

### <--operator-->

#### **$true**

* **$true** is the boolean operator expressing that a condition statement MUST be true.
* **$true** evaluates to true, if
  + its condition evaluates to true.

#### **$false**

* **$false** is the boolean operator expressing that a condition statement MUST be false.
* **$false** evaluates to true, if
  + its condition evaluates to false.

#### **<--operation-->**

* If an operator is used that does not correspond to any of boolean operators (**$true**, **$false**), it is considered to be an XDI operation (e.g., **$get**, **$set**, **$del**, **$do<--extension-->**).
* The XDI operation evaluates to true, if
  + it exists in the input message.

### <--condition-->

#### **{$from} Variable**

* **{$from}** is a global XDI variable representing the sender of the input message (see **<--from-->** in the [XdiMessagePatterns](https://wiki.oasis-open.org/xdi/XdiMessagePatterns) proposal).
* **{$from}** MUST be replaced by the XDI address of the sender of the input message before the condition is evaluated.

#### **{$msg} Variable**

* **{$msg}** is a global XDI variable representing the global context of the input message (see **<--from-->[$msg]!<--id-->** in the [XdiMessagePatterns](https://wiki.oasis-open.org/xdi/XdiMessagePatterns) proposal).
* **{$msg}** MUST be replaced by the XDI address of the input message before the condition is evaluated.

#### **$is**

* **$is** expresses that the subject and object nodes of the condition represent the same logical entity (see the [EquivalenceLinks](https://wiki.oasis-open.org/xdi/EquivalenceLinks) proposal).
* **$is** evaluates to true, if
  + the subject node and object node of the condition represent the same logical entity.

#### **$equals**

* **$equals** expresses that the subject and object nodes of the condition have the same literal values.
* **$equals** evaluates to true, if
  + both the subject node and object node of the condition identify an XDI literal, AND
  + the literal values are equivalent.

#### **$matches**

* **$matches** expresses that the literal value of the subject node of the condition matches the regular expression specified in the literal value of the object node of the condition. [TODO: define which regular expression syntax to use]
* **$matches** evaluates to true, if
  + both the subject node and object node of the condition identify an XDI literal, AND
  + the literal value of the subject node of the condition matches the regular expression specified in the literal value of the object node of the condition.

#### **$greater**

* **$greater** expresses that the literal value of the subject node of the condition must be greater than the literal value of the object node of the condition.
* **$greater** evaluates to true, if
  + both the subject node and object node of the condition identify an XDI literal, AND
  + both literal values are numerical, AND
  + the numerical literal value of the subject node is greater than the numerical literal value of the object node.

#### **$lesser**

* **$lesser** expresses that the literal value of the subject node of the condition must be lesser than the literal value of the object node of the condition.
* **$lesser** evaluates to true, if
  + both the subject node and object node of the condition identify an XDI literal, AND
  + both literal values are numerical, AND
  + the numerical literal value of the subject node is lesser than the numerical literal value of the object node.

#### **<--statement-->**

* If a condition is used that does not correspond to any of comparison conditions (**$is**, **$equals**, **$greater**, **$lesser**), it is considered to be an arbitrary XDI statement.
* The XDI statement evaluates to true, if
  + it exists in the target graph.

#### **$Use**

* **$use** defines the root context node of a usage policy expression subgraph.
* **$use** MUST be a child context of a $get operation in a link contract instance
* **$use** MUST be an entity singleton.
* **$use** MAY have zero or more subcontexts.

# Link Contracts

Per the XDI policy model, all access to graphs is controlled through link contracts. Some of these are general link contracts – such as the root link contract and the public link contract – to set default policies over a broad graph context. Other link contracts enable relationships with specific external authorities (such as an individual person) or generic types of authorities (such as “all customers”).

Link contracts reflecting relationships among authorities are created through a link contract (LC) establishment process. The LC establishment process makes use of link contract templates, governor link contracts and link contract instances.

## Link Contract Establishment Participants

The parties to the establishment of a link contract are:

1. **Requesting authority (RA)**: the XDI authority requesting data and permissions through a link contract.
2. **Authorizing authority (AA)**: the XDI authority authorizing an instance of a link contract in order to grant permissions to data under its authority.
3. **Template authority (TA)**: the XDI authority publishing a link contract template, which may or may not be the requesting authority or authorizing authority.

## Link Contract Pattern Types

There are three core link contract patterns used for different stages of defining an XDI authorization relationship.

1. A **link contract template** is a link contract containing XDI variables that must be replaced by the authorizing authority in order to create a valid link contract instance. A link contract template may be published by any XDI authority, not just a requesting or authorizing authority. Templates provided by neutral third parties make it much easier for link contracts to be standardized, promoting interoperability of XDI vocabulary and permissions. Note: Separating the RA and TA function has not been specified in detailed examples in this working draft or tested in implementation by TC members.
2. A **link contract instance** is produced by replacing the variables in a link contract template.

## A **governor link contract** is a special link contract used by a requesting authority to govern acceptance of link contract instances conforming to a specific link contract template. The governor link contract expresses the policies enforced by a specific requesting authority for accepting instances of a specific link contract template. The address of a governor link contract is algorithmically derived from the address of the link contract template on which it is based, providing a standard way to authorize acceptance of link contract instances based on a known template. The governor link contract also provides a way for the AA to update the link contract instance repeatedly as needed, or to delete the instance.Link Contract Establishment

The LC establishment process describes building block patterns for multiple use cases. As a health warning, the TC has not worked out use cases involving write access to an AA’s graph as much as worked on read access ($get) at the time of this graph and there may be as-yet-unexplored consistency issues with writes.

A spectrum of use cases example for read access for LC establishment involving read access ($get) to the AA’s graph are:

* An RA could request a combination of mandatory and optional data items from an AA.
* It could use and discard those items without ever retrieving the LC instance.
* It could persist just the address of an LC instance (so as to know it has already completed LC establishment in the event of future access needs)
* It could retain the full contents of an LC instance but retain no data (so as to know in advance exactly what its access rights are to an AA but not risk inconsistency or incur compliance obligations associated with storing a copy of the data).
* It can copy or cache the full LC instance and data without regard to usage policies in the absence of a contract with the AA
* It can copy or cache the full LC instance and data subject to usage policies (on a contractual basis)

Link contract establishment involves the following steps:

1. RA sends an XDI message identifying itself to the AA and providing a link contract template (from itself or a TA) describing the requested access.

Link Contract Template Pattern:

<--TA--><--ID-->{$do}/<--operation-->/{$to}<--object-graph-->

1. AA either

Rejects the access request and returns a $AccessDenied error in the message result (sub-contexts TBD) and this process ends, or

AA determines request can be granted by performing all the requested operations (and returning all mandatory data elements for $get operations). AA generates a link contract instance granting the requested access, adds it to its graph, and may return the requested data and link contract instance to the RA in a message result. What is actually returned depends on exactly what was requested by the RA for the use case. Note: examples in the back of this working draft have not been updated to show the various possible $set$do message patterns for the use cases described above).

In the case of success the LC instance pattern is as follows:

Link Contract Instance Pattern:

**(<--AA-->/<--RA-->)<--TA--><--ID-->$do/<--operation-->/(<--AA-->/<--RA-->)<--AA--><--object-graph-->$t**

1. If requested by the LC template, AA algorithmically generates the address and ID of the RA’s governor link contract corresponding to the LC template from Step 1. AA sends a second XDI message (authorized by the governor LC) requesting to add the address of the LC instance from Step 2 to the RA’s graph.

Link Contract Governor Pattern:

**(<--RA-->/<--TA--><--ID-->{$do})$do/$set{$do}/(<--RA-->/<--TA--><--ID-->{$do})({$to}/<--RA-->)<--TA--><--ID-->$do**

1. Depending on the use case, the RA may write the LC instance and the requested data to its graph. RA MUST persist the LC instance if it persists a copy of requested data. RA SHOULD NOT violate data usage directives $DoNotCopy and $ExpireCopy.

AA’s SHOULD sign LC instances and RA’s MUST verify signatures.

Note: LC instances should always be time stamped and for many use cases, should also be signed. This working draft does specify the time stamp. It does not yet define what is the best mechanism for an RA to request, or require signature from the AA. This may be just another data element requested in the template. Examples need to be added to future versions of this working draft.

## Summary of Link Contract Patterns

Following is a high-level summary of the three link contract pattern types.

**Singleton Patterns**

TEMPLATE PATTERN:

<--TA--><--ID-->{$do}/<--operation-->/{$to}<--object-graph-->

GOVERNOR PATTERN:

(<--RA-->/<--TA--><--ID-->{$do})$do/$set{$do}/(<--RA-->/<--TA--><--ID-->{$do})({$to}/<--RA-->)<--TA--><--ID-->$do

GOVERNOR PATTERN short notation:

<--RA-->/<--TA--><--ID-->{$do}/($do/$set{$do}/({$to}/<--RA-->)<--TA--><--ID-->$do)

SPECIFIC INSTANCE:

(<--AA-->/<--RA-->)<--TA--><--ID-->$do/<--operation-->/(<--AA-->/<--RA-->)<--AA--><--object-graph-->

SPECIFIC INSTANCE short notation:

<--AA-->/<--RA-->/(<--TA--><--ID-->$do/<--operation-->/<--AA--><--object-graph-->)

GENERIC INSTANCE:

(<--AA-->/<--class-->)<--TA--><--ID-->$do/<--operation-->/(<--AA-->/<--class-->)<--AA--><--object-graph-->

GENERIC INSTANCE short notation:

<--AA-->/<--class-->/(<--TA--><--ID-->$do/<--operation-->/<--AA--><--object-graph-->)

**Collection Patterns**

TEMPLATE PATTERN:

GOVERNOR PATTERN:

GOVERNOR PATTERN short notation:

SPECIFIC INSTANCE:

(<--AA-->/<--RA-->)<--TA--><--ID-->[$do]!:uuid1234/<--operation-->/(<--AA-->/<--RA-->)<--AA--><--object-graph-->

SPECIFIC INSTANCE short notation:

<--AA-->/<--RA-->/(<--TA--><--ID-->[$do]!:uuid1234/<--operation-->/<--AA--><--object-graph-->)

GENERIC INSTANCE:

(<--AA-->/<--class-->)<--TA--><--ID-->[$do]!:uuid1234/<--operation-->/(<--AA-->/<--class-->)<--AA--><--object-graph-->

GENERIC INSTANCE short notation:

<--AA-->/<--class-->/(<--TA--><--ID-->[$do]!:uuid1234/<--operation-->/<--AA--><--object-graph-->)

Where:

* <--TA--> is the XDI address of the template authority.
* <--RA--> is the XDI address of the requesting authority.
* <--AA--> is the XDI address of the authorizing authority.
* <--ID--> is an OPTIONAL context for specialization of the singleton $do or collection [$do]<--member-id-->. It can serve as a "qualifier" or "tag" that describes the intention of the link contract, e.g. #registration.
* <--member-id--> is a unique immutable ID for a member of a link contract collection. (Unordered members start with !, ordered members start with @.)
* <--operation--> is an XDI operation permitted by this link contract (e.g., **$get**, **$set**, **$del**, **$copy**, **$move**, or **$do<--extension-->**).
* <--object-graph--> is the XDI graph that is the object of the operations permitted by the link contract.
* <--boolean-context--> is defined in the Policy Expression Requirements.
* <--operator--> is defined in the Policy Expression Requirements.
* <--condition--> is defined in the Policy Expression Requirements..

Notes about this overall architecture:

1. The only difference between the singleton and collection patterns is whether the link contract node itself is an XDI entity singleton or entity collection.
2. Both the template and governor patterns begin with XDI variables ({$from} for a template and {$to} for a governor) because they deal with link contracts containing XDI variables outside of the policy expression branch. The instance pattern does not include XDI variables because they have been instantiated in order to create the instance.
3. In the instance pattern, each of the three components specializes the associated $word that follows it, i.e.:
   1. <--authorizing-authority--> specializes $to.
   2. <--requesting-authority--> specializes $from.
   3. <--template-authority--><--template-id--> specializes $do.

## Link Contract Template Pattern

All link contract templates MUST consist of a set of XDI statements in one of the two following patterns:

**Link Contract Template Singleton Pattern**

<--template-authority-->{$from}<--template-authority--><--template-id-->$do/<--operation-->/{$to}<--object-graph-->  
<--template-authority-->{$from}<--template-authority--><--template-id-->$do$if<--boolean-context-->/<--operator-->/<--condition-->

**Link Contract Template Collection Pattern**

<--template-authority-->{$from}<--template-authority--><--template-id-->[$do]<--member-id-->/<--operation-->/{$to}<--object-graph-->  
<--template-authority-->{$from}<--template-authority--><--template-id-->[$do]<--member-id-->$if<--boolean-context-->/<--operator-->/<--condition-->

### The {$from} Variable

To define a link contract template, the XDI address of a link contract singleton or collection is placed in the context of a {$from} variable representing the requesting authority. This is the same variable used to represent the sender of an XDI message in [XDI messaging](https://wiki.oasis-open.org/xdi/XdiMessagePatterns). Note that this role reflects the directionality of the requested access rather than the communication of the request; i.e., it is the same whether the requesting authority is actively sending the link contract template to an authorizing authority in an XDI $set{$do} message or passively accepting requests for the link contract template available via an XDI $get message.

### The {$to} Variable

The {$to} variable represents the authorizing authority who will authorize a link contract instance. This is the same variable used to represent the receiver of an XDI message in [XDI messaging](https://wiki.oasis-open.org/xdi/XdiMessagePatterns).

### Link Contract Permissions

1. Each relational <--operation--> predicate in the $do or [$do]<--member-id--> context MUST define a permission granted by the link contract to an XDI message that satisfies the link contract policies.
2. The <--object-graph--> of each <--operation--> predicate defines the XDI graph or subgraph over which the permission for the specified operation is granted.

### Link Contract Policies

A link contract MAY contain one or more $do$if or [$do]<--member-id-->$if statements. These are the policy expression branches of the link contract. They express the policies that an authorizing authority will apply to allow or deny XDI messages that claim to be authorized under the link contract. XDI policies can express any set of rules that can be captured using XDI policy expression syntax.

## Governor Link Contract Pattern

A governor link contract is used by a requesting authority to authorize acceptance of link contract instances based on a link contract template. IMPORTANT: the XDI address of the governor link contract is algorithmically composed by placing the XDI address of the link contract template inside the XDI address of the requesting authority followed by the $from context. This enables XDI clients to reference the governor link contract for authorization in an XDI $set{$do} message for a new link contract instance (see [XDI messaging](https://wiki.oasis-open.org/xdi/XdiMessagePatterns)).

All governor link contracts except a root governor link contract (see below) MUST consist of a set of XDI statements in one of the two following patterns:

**Governor Link Contract Singleton Pattern**

<--requesting-authority-->{$to}<--template-authority--><--template-id-->$do/$set{$do}/{$from}<--template-id-->$do  
<--requesting-authority-->{$to}<--template-authority--><--template-id-->$do$if<--boolean-context-->/<--operator-->/<--condition-->

**Governor Link Contract Collection Pattern**

<--requesting-authority-->{$to}<--template-authority--><--template-id-->[$do]<--member-id-->/$set{$do}/{$from}<--template-id-->[$do]<--member-id-->  
<--requesting-authority-->{$to}<--template-authority--><--template-id-->[$do]<--member-id-->$if<--boolean-context-->/<--operator-->/<--condition-->

### $set{$do} Operation

A governor link contract uses the $set{$do} operation that is used to authorize new link contract instances. The object of this predicate MUST be the XDI address of the link contract template for which it will authorize instances.

### Governor Link Contract Policies

The [XDI policy expression](https://wiki.oasis-open.org/xdi/XdiPolicyExpression) statements in a governor link contract expresses the requesting authority's policy for accepting instances of a specific link contract template. For example, these policy statements might require the authorizing authority to:

* Be an employee of a company.
* Be under or over a certain age.
* Have a certain reputation or trust level.

## Link Contract Instance Pattern

All link contract instances except a root link contract or public link contract (see below) MUST consist of a set of XDI statements in one of the two following patterns:

**Link Contract Instance Singleton Pattern**

<--authorizing-authority-->$to<--requesting-authority-->$from<--template-authority--><--template-id-->$do/<--operation-->/<--authorizing-authority--><--object-graph-->  
<--authorizing-authority-->$to<--requesting-authority-->$from<--template-authority--><--template-id-->$do$if<--boolean-context-->/<--operator-->/<--condition-->

**Link Contract Instance Collection Pattern**

<--authorizing-authority-->$to<--requesting-authority-->$from<--template-authority--><--template-id-->[$do]<--member-id-->/<--operation-->/<--authorizing-authority--><--object-graph-->  
<--authorizing-authority-->$to<--requesting-authority-->$from<--template-authority--><--template-id-->[$do]<--member-id-->$if<--boolean-context-->/<--operator-->/<--condition-->

Note that in link contract instances, all XDI variables have been replaced.

## Root Link Contract

The root link contract is the bootstrap link contract that gives an XDI authority access to its own XDI subgraph.

1. The root link contract is based on the generic link contract instance pattern, with the following constraints:
   1. The requesting authority is set to the XDI authority.
   2. The authorizing authority is set to the XDI authority.
   3. The template ID is empty.
   4. The root link contract MUST use the singleton pattern, i.e. there MUST only be one per XDI authority.
2. The root link contract SHOULD grant $all permission to the XDI authority's subgraph.
3. The root link contract SHOULD contain a policy that restricts the sender of an XDI message to the XDI authority's subgraph.
4. Authentication SHOULD be required for access to a root link contract. However the XDI authority is the ultimate authority for the policies pertaining to any link contract.

**Root Link Contract Pattern**

<--authorizing-authority-->$to<--authorizing-authority-->$from$do/$all/<--authorizing-authority-->  
<--authorizing-authority-->$to<--authorizing-authority-->$from$do$if/$true/({$from}/$is/<--authorizing-authority-->)

## Public Link Contract

The public link contract is the well-known link contract that an XDI authority can offer for access to the portion of its XDI subgraph (if any) that is publicly available.

1. The public link contract is based on the generic link contract instance pattern, with the following constraints:
   1. The requesting authority is set to $anon.
   2. The authorizing authority is set to the XDI authority.
   3. The template ID is set to $public.
   4. The public link contract MUST use the singleton pattern, i.e. there MUST only be one per XDI authority.
2. The public link contract SHOULD grant $get permission to the $public subgraph of the XDI authority's subgraph.
3. In addition, the public link contract MAY grant permissions to other parts of the XDI authority's subgraph.
4. The public link contract SHOULD NOT contain a policy.
5. Authentication SHOULD NOT be required for access to a public link contract. However the XDI authority is the ultimate authority for the policies pertaining to any link contract.

**Public Link Contract Pattern**

<--authorizing-authority-->$to$anon$from$public$do/$get/<--authorizing-authority-->$public

# Usage Policy Expression Dictionary Requirements

The following usage policies are defined by an AA to define what can be done with data after it is retrieved via a $get operation.

* $DoNotCopy – Enables AA to demand that data returned in a message result for a $get operation be discarded after it is retrieved and used by the RA. This expression SHOULD NOT be used unless authorities are bound by a legal contract (such as a trust framework) to follow the directive.
* $ExpireCopy$t – Enables AA to demand RA set an expiration data/time on any copy or cache of the data retrieved in the message result for a $get operation. RAs SHOULD honor the expire copy directive.

The following usage policies can be requested by an RA from an AA:

* $OptionalElement – Enables RA to indicate that one of the data items requested in a $get operation is optional enabling a link contract instance to be established without necessarily including that data item. If this context is not present, the requested data is assumed to be required and the AA MUST either return or reject the data and establish an LC instance accordingly.
* Note for future work only: $Subscription – Enables RA to indicated that it intends to copy the entire results of a $get operation and requests that the AA update the information in the RA’s graph if the information changes.

# Policy Expression Examples

These examples are written in [XDI Display Format](https://wiki.oasis-open.org/xdi/XdiDisplayFormat). See [JSON Serialization Rules](https://wiki.oasis-open.org/xdi/JSONSerializationRules) for the over-the-wire format.

Link contract policy that checks that the secret token in the message matches the secret token in the graph:

$secret<$token>&/&/"s3cr3t"  
$do$if/$true/({$msg}$secret<$token>/$equals/$secret<$token>)

Link contract policy that checks that the message timestamp comes before a certain time:

$do$if/$true/({$msg}<$t>/$lesser/...$do#expiration<$t>)  
$do#expiration<$t>&/&/"2012-12-12"

Link contract policy that checks that the message comes from one of the approved senders:

$do$if/$true/({$from}/$is/[=]!2222)  
$do$if/$true/({$from}/$is/[=]!3333)  
$do$if/$true/({$from}/$is/[=]!4444)

Link contract policy that excludes a certain sender:

$do$if/$false/({$from}/$is/[=]!1111)

Link contract policy that excludes a certain sender (other way of doing it):

$do$if$not/$true/({$from}/$is/[=]!1111)

Link contract for "root access" to an authority's own graph:

$secret<$token>&/&/"s3cr3t"  
$do$if$and/$true/({$from}/$is/[=]!1111)  
$do$if$and/$true/({$msg}$secret<$token>/$equals/$secret<$token>)  
$do/$all/()

Link contract for the "public branch" of an XDI graph:

$public$do/$get/$public

Link contract to allow Bob access to Alice's phone number:

=alice/$ref/[=]!1111  
=bob/$ref/[=]!2222  
[=]!1111$<#tel>&/&/"#1-123-567-8900"  
$do$if/$true/({$from}/$is/[=]!2222)  
$do/$get/[=]!1111<#tel>

Link contract policy that checks that the message comes either from myself ([=]!1111) or from one of my friends:

$do$if$or/$true/([=]!1111/$is/{$from})  
$do$if$or/$true/([=]!1111/#friend/{$from})

Link contract that allows a $do$signal operation only on certain statements (with variables):

$do/$do$signal/[=]!1111[#channel]!23[#event]  
$do$if$and/$do$signal/([=]!1111[#channel]!23[#event]{1}/#domain/#cloudos)  
$do$if$and$or/$do$signal/([=]!1111[#channel]!23[#event]{1}/#type/#notification)  
$do$if$and$or/$do$signal/([=]!1111[#channel]!23[#event]{1}/#type/#subscription)  
$do$if$and$or/$do$signal/([=]!1111[#channel]!23[#event]{1}/#type/#deletion)

Combination of the previous two examples:

[=]!1111/#friend/[=]!2222  
$do/$do$signal/[=]!1111[#channel]!23[#event]  
$do$if$and[$or]!2/$true/([=]!1111/$is/{$from})  
$do$if$and[$or]!2/$true/([=]!1111/#friend/{$from})  
$do$if$and/$do$signal/([=]!1111[#channel]!23[#event]{1}/#domain/#cloudos)  
$do$if$and[$or]!1/$do$signal/([=]!1111[#channel]!23[#event]{1}/#type/#notification)  
$do$if$and[$or]!1/$do$signal/([=]!1111[#channel]!23[#event]{1}/#type/#subscription)  
$do$if$and[$or]!1/$do$signal/([=]!1111[#channel]!23[#event]{1}/#type/#deletion)

Link contract policy that allows the message if it is not sent on Friday 13th, and it is not sent by =drummond or =markus. (Note that the weekday and day-of-month definitions still need to be defined in the XDI $ Dictionary.)

$do$if$and[$not]!1$and/$true/[weekday\_is\_friday]  
$do$if$and[$not]!1$and/$true/[day\_is\_13th]  
$do$if$and[$not]!2$or/$true/(=drummond/$is/{$from})  
$do$if$and[$not]!2$or/$true/(=markus/$is/{$from})

# Link Contract Examples

These examples are written in [XDI Display Format](https://wiki.oasis-open.org/xdi/XdiDisplayFormat), with line breaks added after each predicate for readability and to add comments. See [JSON Serialization Rules](https://wiki.oasis-open.org/xdi/JSONSerializationRules) for the over-the-wire format.

## Public Link Contract for XDI Discovery

In this XDI graph, the public link contract provides access to the endpoint URIs in the XDI graph for the authority [=]!:uuid:2 to support [XDI Discovery](https://wiki.oasis-open.org/xdi/XdiDiscovery).

**NOTE: TO AVOID CONFUSION, MAKE SURE YOUR BROWSER WINDOW IS WIDE ENOUGH SO LINES DO NOT WRAP.**

/$is$ref/([=]!:uuid:2) <== inverse statement of peer root address for the XDI authority  
([=]!:uuid:2)/$ref/ <== direct statement of peer root address for the XDI authority  
[<$uri>]<!1>&/&/"https://xdi.example.com/1224/" <== endpoint URI 1  
[<$uri>]<!2>&/&/"https://xdi2.example.com/1224/" <== endpoint URI 2  
<$uri>/$ref/[<$uri>]<!1> <== reference to URI 1 as default endpoint URI  
[=]!:uuid:2$to$anon$from$public$do/$get/<$uri> <== public link contract to get reference to default endpoint URI  
[=]!:uuid:2$to$anon$from$public$do/$get/[<$uri>] <== public link contract to get collection of all endpoint URIs

### Alice-to-Acme Link Contract Pattern

This set of XDI graphs shows how Acme Corp can publish a link contract template requesting a name and email address, and Alice can authorize an instance of that link contract template. **For readability, all UUIDs are represented by 1 digit.**

**NOTE: TO AVOID CONFUSION, MAKE SURE YOUR BROWSER WINDOW IS WIDE ENOUGH SO LINES DO NOT WRAP.**

#### Acme's Starting Graph

This is Acme's graph publishing Acme's own link contract template and governor link contract before Alice has authorized an instance of this template.

()/$is$ref/ <== inverse statement of peer root address for the XDI authority  
 ([+]!:uuid:1)  
([+]!:uuid:1)/$ref/ <== direct statement of peer root address for the XDI authority  
 ()  
+acme/$ref/ <== name-to-number ref  
 [+]!:uuid:1  
{$from}[+]!:uuid:1#registration$do/$get/ <== link contract template statement requesting a default name  
 {$to}<#name>  
{$from}[+]!:uuid:1#registration$do/$get/ <== link contract template statement requesting a default phone number  
 {$to}<#tel>  
{$from}[+]!:uuid:1#registration$do$if$and/$true/ <== link contract template policy requiring the requester to be Acme  
 ({$from}/$is/[+]!:uuid:1)  
{$from}[+]!:uuid:1#registration$do$if$and/$true/ <== link contract template policy requiring the request message to be signed  
 ({$msg}<$sig><$valid>&/&/true)  
{$to}[+]!:uuid:1$from[+]!:uuid:1#registration$do/$set{$do}/ <== governor link contract for the link contract template above  
 {$from}[+]!:uuid:1#registration$do  
{$to}[+]!:uuid:1$from[+]!:uuid:1#registration$do$if/$true/ <== governor link contract template policy requiring the instance message to be signed  
 ({$msg}<$sig><$valid>&/&/true)

#### Alice's Starting Graph

This is Alice's graph before Alice authorizes an instance of Acme's registration link contract template.

()/$is$ref/ <== inverse statement of peer root address for the XDI authority  
 ([=]!:uuid:2)  
([=]!:uuid:2)/$ref/ <== direct statement of peer root address for the XDI authority  
 ()  
=alice/$ref/ <== name-to-number ref  
 [=]!:uuid:2  
=]!:uuid:2[<#name>]<!:uuid:3>&/&/ <== name instance UUID 3  
 "Alice"  
=]!:uuid:2<#name>/$ref/ <== reference to name instance UUID 3 as default name  
 =]!:uuid:2[<#name>]<!:uuid:3>  
[=]!:uuid:2[<#tel>]<!:uuid:4>&/&/ <== telephone number instance UUID 4  
 "+1-999-555-1111"  
[=]!:uuid:2[<#tel>]<!:uuid:5>&/&/ <== telephone number instance UUID 5  
 "+1-999-555-2222"  
[=]!:uuid:2[<#tel>]<!:uuid:6>&/&/ <== telephone number instance UUID 6  
 "+1-999-555-3333"  
[=]!:uuid:2<#tel>/$ref/ <== reference to tel instance UUID 4 as default tel  
 [=]!:uuid:2[<#tel>]<!:uuid:4>  
[=]!:uuid:2#mobile<#tel>/$ref/ <== reference to tel instance UUID 4 as mobile tel  
 [=]!:uuid:2[<#tel>]<!:uuid:4>  
[=]!:uuid:2#home<#tel>/$ref/ <== reference to tel instance UUID 5 as home tel  
 [=]!:uuid:2[<#tel>]<!:uuid:5>  
[=]!:uuid:2#work<#tel>/$ref/ <== reference to tel instance UUID 6 as work tel  
 [=]!:uuid:2[<#tel>]<!:uuid:6>

#### New Statements Added to Both Graphs After Link Contract Authorization

Following are the new statements added to both Alice's and Acme's graphs after Alice authorizes an instance of Acme's registration link contract template.

[=]!:uuid:2$to[+]!:uuid:1$from[+]!:uuid:1#registration$do/$get  
 =]!:uuid:2<#name>  
[=]!:uuid:2$to[+]!:uuid:1$from[+]!:uuid:1#registration$do/$get  
 =]!:uuid:2<#tel>

Note that *exactly the same statements* are added to both Alice's and Acme's graphs. They are globally unique in both graphs because:

1. The authorizing authority context is Alice's XDI address.
2. The requesting authority context is Acme's XDI address.
3. The link contract template ID is an entity singleton assigned by Acme under its own XDI address.

# Data Usage Policy Examples

=markus<#email>&/&/"...."

=markus$to=hubert$from$do/$get/=markus<#email>

=markus$to=hubert$from$do$if$and... { you have to be hubert } <-- link contract policy

=markus$to=hubert$from$do$if$and... { you have to send a valid password ...} <-- link contract policy

$use$if$and... { you have to encrypt the data } <-- data usage policy

$use$if$and$not... { you may not cache the data } <-- data usage policy

1. Data usage policy part of the data:

=markus<#email>&/&/"...."

=markus<#email>$use$if$and... { you have to encrypt the data } <-- data usage policy

=markus<#email>$use$if$and$not... { you may not cache the data } <-- data usage policy

2. Data usage policy part of the link contract:

=markus$to=hubert$from$do/$get/=markus<#email>

=markus$to=hubert$from$do$use$if$and... { you have to encrypt the data } <-- data usage policy

=markus$to=hubert$from$do$use$if$and$not... { you may not cache the data } <-- data usage policy

=markus$to=hubert$from$do$if$and... { you have to be hubert } <-- link contract policy

=markus$to=hubert$from$do$if$and... { you have to send a valid password ...} <-- link contract policy

Example message sent by Hubert:

=hubert[$msg]!:uuid:1234$do/$get/=markus<#email>

=hubert[$msg]!:uuid:1234/$do/=markus$to=hubert$from$do