

How to encode MIAME in MAGE

Introduction

The Minimum Information About a Microarray Experiment, also known as MIAME, was developed to specify which microarray experiment data and metadata should be reported to enable others to understand and interpret the experiment unambiguously. This is a data content standard, not a format standard.

Microarray Gene Expression Markup Language (MAGE-ML) is a formal language designed to describe and communicate information about microarray based experiments. MAGE-ML is an XML language, it can be used to describe microarray designs, microarray manufacturing information, microarray experiment setup and execution information, gene expression data and data analysis results.

MAGE-ML has been automatically derived from Microarray Gene Expression Object Model (MAGE-OM), which is developed and described using the Unified Modelling Language (UML) – a standard language for describing object models. Models described using UML have advantages over pure XML technologies (DTDs or XML Schemas) in many respects, especially for didactic purposes. They use graphical representation depicting the relationships between different entities in a way which is much easier to follow for a human than DTDs. The idea behind UML diagrams is to provide a way of describe models that is both human readable and has strict semantics, while DTDs and XML Schemas are meant primarily for computers. Also, complex models (also MAGE) involve many different types of relationships between model elements, while in XML by definition information is encoded in a hierarchical manner and relationships that break the hierarchy need to be encoded in some special ways.

MIAME requires detailed annotation about experimental conditions, materials and procedures to be captured. MAGE-ML is a rich format, by using it one can encode MIAME-required information and more. The purpose of this document is to provide guidance for encoding MIAME-required information in MAGE-ML.

MIAME consists of 5 sections, we will follow that structure here. For each MIAME section the following is presented:

- 1) one or more UML class diagrams containing a subset of classes and associations from the corresponding MAGE-OM package(s) needed for MIAME-compliant data encoding;
- 2) a fragment of the simplified MAGE-ML DTD (we will call this here MAGE-ML-Lite) that is sufficient for encoding MIAME;
- 3) an informal object diagram that illustrates the structure needed for MIAME encoding in MAGE objects;
- 4) a sample MAGE-ML document template that corresponds to the object diagram.

On the class diagrams we have only deleted some classes and associations that are less relevant for MIAME encoding, but we haven't made any structural changes. In fact, the diagram layout is the same as in the formal MAGE-OM specification.

We have simplified MAGE-ML DTD by omitting some elements, attributes and parent-child relations. Still, a document that satisfies MAGE-ML-Lite would also satisfy the MAGE-ML

DTD, i.e., MAGE-ML-Lite is a proper sublanguage of MAGE-ML. We also didn't use entities that are widely used in MAGE-ML DTD for encoding class hierarchies. Note that we are not trying to invent another standard, the purpose is simply to make it easier to learn MAGE-OM and MAGE-ML. Also, MAGE-ML-Lite DTDs are not proper DTDs, ellipsis ("..") is used in places where some elements or attributes have been omitted. This is to emphasise that MAGE-ML-Lite DTD is mostly for educational purposes and not for software construction.

In some cases MAGE-ML-Lite as discussed here might not be sufficient for describing some essential information. Then the full power of MAGE-ML should be used, by using structures not covered here. Therefore we also do not recommend, e.g., building tools that can deal only with MAGE-ML-Lite and break down when confronted with full MAGE-ML; instead such tools should be able to ignore MAGE-ML structures that are not part of MAGE-ML-Lite.

Class models provided are valid UML models. For a brief introduction to UML notation, see <http://www.ajug.org/info/tech/uml/uml.html>

The difference between a class model and an object model is that in the former boxes represent classes that can be used in different situations, while in the latter boxes represent objects (or sets of similarly structured objects) arranged and linked in a particular way. This can be better explained on the basis of "Samples used, extract preparation and labeling" diagrams; in the class model there are BioMaterial, Treatment and BioMaterialMeasurement classes that have associations to each other and form a recursive structure (a BioMaterial has Treatments, each of the Treatments has BioMaterialMeasurements, each of the BioMaterialMeasurement points to a BioMaterial which again can have Treatments etc.). On the object model we show a particular realisation of this structure, with 4 BioMaterial boxes (which are actually called BioSource, BioSample (2 of them), and LabeledExtract, because all these are subclasses of the general BioMaterial class).

The following notation has been used in our object models:

- Shaded boxes with solid outline represent mandatory elements, while white boxes with dashed outline represent optional elements; there are some white boxes with solid outlines, for situations where some optional elements are shown, but if they are used, then some other objects (these white with solid outline ones) must be used as well.
- Object's class (the first line) and attributes (the subsequent lines) are written inside its box. Note that we omit "identifier" attribute that is mandatory for all the main object types which can be referenced from another place in the XML tree or from other documents (these are called "Identifiable" objects in MAGE).
- Thick lines represent parent-child relationships in MAGE-ML (correspond to aggregate associations in MAGE-OM), while thin lines represent references (non-hierarchical relationships) in MAGE-ML.
- Arrows for thick lines point from a child to its parent, while for thin lines they are used to indicate navigability - if object A points to object B, object A knows about object B, while B doesn't have to know about A. This directly corresponds to UML notation. If both objects have to know about each other, no arrows are used on that line.
- By stacks of boxes we are trying to indicate multiple objects, although simple graphical notation is often ambiguous for this purpose. If a child object (i.e., having an outgoing thick arrow) is a stack rather than a simple box, it means that this actually represent multiple child

objects that the parent object can have. If an object does not have parents and is represented by a stack, then the box represents multiple objects that can be referenced via an incoming reference type association (or, if there are multiple such association, then via the "main" one in some sense - this is the source of ambiguity for our object notation, but it doesn't occur too often, and examining DTD which is unambiguous will resolve the issue). Note that in some cases we indicate on the object diagram just a single object where MAGE-OM and MAGE-ML allow multiple objects; this happens when we think that in typical cases a single object is enough.

- OntologyEntry objects are denoted by "OE", and we do not show attributes for these. See a document developed by the MGED Ontologies working group describing how OntologyEntries should be used (see <http://mged.sourceforge.net/ontologies/>, section "Policies for using the MGED Ontology in MAGE documents").

In MAGE-ML examples some colour coding has been used, and comments are either in blue or in red. Optional elements are on a shaded background. ADF (referenced in examples) stands for "Array Description Format" designed by EBI as a simple, tab-delimited array design presentation and exchange format, see

<http://www.ebi.ac.uk/miamexpress/help/adf/ArrayIntroduction.htm> for more information.

All terms (values and category) provided in MAGE-ML examples are based on MGED ontology release 1.1.7. For complete information on MGED ontology see

<http://mged.sourceforge.net/ontologies/index.php>

The rest of this document is MIAME merged with explanations of MAGE-ML encoding. MIAME text is in italics. We have swapped the 2 main MIAME sections, here the experiment description comes first followed by the array design description. We provide a separate document where, for each of the MIAME sections, the 2 above-mentioned diagrams are provided on a separate page. Similarly, DTD fragments are on separate pages, as well as XML example fragments. The following pages are provided:

- Experimental design - diagrams, DTD, examples
- Samples used, extract preparation and labeling - diagrams, DTD, examples
- Hybridization procedures and parameters; measurement data - diagrams, DTD, examples
- Measurement data and specifications of data processing - diagrams, DTD, examples
- Array design description (1) - diagrams, DTD, examples
- Array design description (2) - diagrams, DTD, examples
- Other MAGE structures (common constructs used in other sections):
 - Miscellaneous: quantitation types - class diagram, DTD, example
 - Miscellaneous: protocols - class diagrams, DTD, example
 - Miscellaneous: descriptions and contacts - class diagrams, DTD, examples

Besides these we also provide:

- complete MAGE-ML-Lite DTD where elements are ordered in the same way as in MAGE-ML DTD, to ease comparison (a separate document).

- MAGE-ML-Lite DTD fragments on package-by-package basis, like in the above-mentioned pages, but with comments retained (at the end of this document).
- The same set of examples as used in the supplementary document, but as stand-alone files, archived. In the same archive we have also including diagrams – another representation of MAGE-ML object structure. See readme file also included in the archive for details on these diagrams.

Identifiers in MAGE-ML

The object identifiers for MAGE objects should have the form

`<authority>:[<namespace>]:<object>[:<revision>]`

where ":" is the field separator, "<...>" is a string and "[...]" represents an optional elements. Syntactically none of strings used in 'authority', 'namespace', 'object' and 'revision' is allowed to contain ':'.

For the time being `<authority>:<object>` also is acceptable from current MAGE-ML exporters, but we would recommend that submitters strive to conform to the specified format with null namespace, e.g. `<authority>::<object>`. If submitters don't have a meaningful namespace then the recommended format is `<authority>::<object>`.

`<authority>` is assigned by the data provider. We recommend that this is done in a way that minimises the possibility of clashes, for instance following the DNS model with the providers giving names like "ebi.ac.uk", "umich.edu", "genetics.umich.edu" or "lab23.genetics.umich.edu"

We recommend that the software manufacturers include the assignment of the authority during the installation of each particular copy of their software. The defaults should be set in a way that minimise the clashes.

If there is a 'authority' clash during the submission to a public database (e.g., ArrayExpress), we will try to resolve this via MGED. The mass software should ideally have an option which allows to change the 'authority' after the installation.

Regarding `[<namespace>]:<object>[:<revision>]`, the only requirement at the moment is that objects should be guaranteed to be unique within the authority.

The following table shows an example of MAGE-ids implementation in MIAMExpress tool. Note the way the “Namespaces” are formed.

Identifier	Former MiamExpress Prefixes	MAGE-approved Identifiers:
Array	AR	ebi.ac.uk:MIAMExpress/Array:<object>
ArrayManufacture	ARMANUF	ebi.ac.uk:MIAMExpress/ArrayManufacture:<object>
ArrayDesign	_	ebi.ac.uk:MIAMExpress/ArrayDesign:<object>
PhysicalArrayDesign	A-MEXP-#	ebi.ac.uk:MIAMExpress/PhysicalArrayDesign:A-MEXP-#
Zone	ARZ	ebi.ac.uk:MIAMExpress/Zone:<object>
Feature	F	ebi.ac.uk:MIAMExpress/Feature:<object>
FeatureGroup	FG	ebi.ac.uk:MIAMExpress/FeatureGroup:<object>

ReporterGroup	RG	ebi.ac.uk:MIAMExpress/ReporterGroup:<object>
CompositeGroup	CG	ebi.ac.uk:MIAMExpress/ArrayDesign:<object>
Organization	ORG	ebi.ac.uk:MIAMExpress/Organisation:<object>
Person	PERS	ebi.ac.uk:MIAMExpress/Person:<object>
Security	_	ebi.ac.uk:MIAMExpress/Security:<object>
SecurityGroup	_	ebi.ac.uk:MIAMExpress/SecurityGroup:<object>
Channel	Channel	ebi.ac.uk:MIAMExpress/Channel:<object>
PhysicalBioAssay	PBA	ebi.ac.uk:MIAMExpress/PhysicalBioAssay:<object>
MeasuredBioAssay	MBA	ebi.ac.uk:MIAMExpress/MeasuredBioAssay:<object>
DerivedBioAssay	DBA	ebi.ac.uk:MIAMExpress/DerivedBioAssay:<object>
Image	IMG	ebi.ac.uk:MIAMExpress/Image:<object>
MeasuredBioAssayData	DATA	ebi.ac.uk:MIAMExpress/MeasuredBioAssayData:<object>
DerivedBioAssayData	DATA	ebi.ac.uk:MIAMExpress/DerivedBioAssayData:<object>
BioAssayDimension	BAD	ebi.ac.uk:MIAMExpress/BioAssayDimension:<object>
FeatureDimension	DED	ebi.ac.uk:MIAMExpress/FeatureDimension:<object>
ReporterDimension	DED	ebi.ac.uk:MIAMExpress/ReporterDimension:<object>
CompositeSequenceDimension	DED	ebi.ac.uk:MIAMExpress/CompositeSequenceDimension:<object>
QuantitationTypeDimension	QTD	ebi.ac.uk:MIAMExpress/QuantitationTypeDimension:<object>
Treatment	T	ebi.ac.uk:MIAMExpress/Treatment:<object>
BioAssayTreatment	_	ebi.ac.uk:MIAMExpress/BioAssayTreatment:<object>
BioAssayCreation	PBAC	ebi.ac.uk:MIAMExpress/BioAssayCreation:<object>
Hybridization	HYB	ebi.ac.uk:MIAMExpress/Hybridization:<object>
ImageAcquisition	PBASCAN	ebi.ac.uk:MIAMExpress/ImageAcquisition:<object>
FeatureExtraction	FEXT	ebi.ac.uk:MIAMExpress/FeatureExtraction:<object>
Transformation	TFM	ebi.ac.uk:MIAMExpress/Transformation:<object>
BioAssayMap	BAM	ebi.ac.uk:MIAMExpress/BioAssayMap:<object>
QuantitationTypeMap	QTM	ebi.ac.uk:MIAMExpress/QuantitationTypeMap:<object>
FeatureReporterMap	FRM	ebi.ac.uk:MIAMExpress/FeatureReporterMap:<object>
ReporterCompositeMap	RCM	ebi.ac.uk:MIAMExpress/ReporterCompositeMap:<object>
CompositeCompositeMap	CCM	ebi.ac.uk:MIAMExpress/CompositeCompositeMap:<object>
Compound	COMP	ebi.ac.uk:MIAMExpress/Compound:<object>
BioSource	S	ebi.ac.uk:MIAMExpress/BioSource:<object>
BioSample	S	ebi.ac.uk:MIAMExpress/BioSample:<object>
LabeledExtract	S	ebi.ac.uk:MIAMExpress/LabeledExtract:<object>
Treatment	T	ebi.ac.uk:MIAMExpress/Treatment:<object>
BioSequence	BS	ebi.ac.uk:MIAMExpress/BioSequence:<object>
Database	DB	ebi.ac.uk:MIAMExpress/Database:<object>
CompositeSequence	CS	ebi.ac.uk:MIAMExpress/CompositeSequence:<object>
Reporter	R	ebi.ac.uk:MIAMExpress/Reporter:<object>
Experiment	E-MEXP-#	ebi.ac.uk:MIAMExpress/Experiment:E-MEXP-#
ExperimentalFactor	EF	ebi.ac.uk:MIAMExpress/ExperimentalFactor:<object>
FactorValue	EFV	ebi.ac.uk:MIAMExpress/FactorValue:<object>
BioAssayDataCluster	_	ebi.ac.uk:MIAMExpress/BioAssayDataCluster:<object>
Hardware	HW	ebi.ac.uk:MIAMExpress/Hardware:<object>
Parameter	PARAM	ebi.ac.uk:MIAMExpress/Parameter:<object>
Protocol	P-MEXP-#	ebi.ac.uk:MIAMExpress/Protocol:P-MEXP-#
Software	SW	ebi.ac.uk:MIAMExpress/Software:<object>
DerivedSignal	QT	ebi.ac.uk:MIAMExpress/DerivedSignal:<object>

Error	QT	ebi.ac.uk:MIAMExpress/Error:<object>
MeasuredSignal	QT	ebi.ac.uk:MIAMExpress/MeasuredSignal:<object>
PresentAbsent	QT	ebi.ac.uk:MIAMExpress/PresentAbsent:<object>
PValue	QT	ebi.ac.uk:MIAMExpress/PValue:<object>
Ratio	QT	ebi.ac.uk:MIAMExpress/Ratio:<object>
SpecializedQuantitationType	QT	ebi.ac.uk:MIAMExpress/SpecializedQuantitationType:<object>

Experimental design

This section is common to all the hybridizations done in the experiment, such as the goal, brief description, experimental factors tested. It includes the following.

1) Authors, laboratory, contact

Person and Organization are the two subclasses of Contact class. One or more Person objects can be attached to the Experiment, and each of them should contain a reference to an Organization object. Suggested minimal attributes are name, address, phone and email for a Person, and name, URI and address for an Organization.

2) Type of the experiment, for instance,

- *normal vs. diseased comparison*
- *treated vs. untreated comparison*
- *time course*
- *dose response*
- *effect of gene knock-out*
- *effect of gene knock-in (transgenics)*
(multiple types possible)

An ExperimentDesign object has to be attached to the Experiment object, and experiment type is provided in an OntologyEntry object associated to the ExperimentDesign. There can be more than one type.

3) Experimental factors, i.e. parameters or conditions tested, for instance,

- *time*
- *dose*
- *genetic variation*
- *response to a treatment or compound*
(also, see <http://www.mged.org/ontology>)

ExperimentalFactors are attached to the ExperimentDesign object, and each of them must have a category (an OntologyEntry). Each of the ExperimentalFactors in turn has one or more FactorValue objects attached to it (described below in more detail).

4) How many hybridizations in the experiment?

This is not explicitly provided in MAGE, but an Experiment has to be linked to all BioAssays (PhysicalBioAssays, MeasuredBioAssays and DerivedBioAssays) performed, as described below, and the number of hybridizations is the number of PhysicalBioAssays that have associated Hybridizations (there are also other PhysicalBioAssays - see below).

5) *If a common reference is used for all the hybridizations?*

6) *Quality control steps taken:*

- *if any replicates done (yes/no), what type of replicates, description?*
- *whether dye swap is used (only for two channel platforms)?*
- *other (e.g., polyA tails, low complexity regions, unspecific binding)*

There are two types of Description objects that can be attached to ExperimentDesign, a quality control Description and a replicate Description. These can be used for free-text descriptions, however, ontology entries can be attached if it is important to capture quality control information in a more structured way (e.g., enabling queries).

5) *A brief description of the experiment and its goal and a link to a publication if exists*

6) *Links (URL), citations*

A Description object should be attached to the Experiment, it can contain a URL pointing to additional information. BibliographicReference objects can be attached to this Description, providing publication details. The publication type must be provided using an associated OntologyEntry.

Samples used, extract preparation and labeling

By a sample we understand the biological material (biomaterial), from which the nucleic acids have been extracted for subsequent labeling and hybridization. In this section all steps that precedes the hybridization with the array are described. We can usually distinguish between the source of the sample (bio-source, e.g., organism, cell type or line), its treatment, the extract preparation, and its labeling. MGED is developing an ontology for sample description (see <http://www.mged.org/ontology>) the use of which is encouraged. Here we list the most essential items that are usually needed.

In MAGE there is an abstract class BioMaterial and three subclasses, BioSource, BioSample and LabeledExtract. BioSource corresponds to the source of the sample in MIAME, a LabeledExtract is what is hybridized to an array, and BioSamples correspond to any intermediate materials in the sample processing chain.

1) *Bio-source properties*

- *organism (NCBI taxonomy)*

Encoded as a characteristic (OntologyEntry) of the BioSource

- *contact details for sample*

A Person object can be linked from a BioSource, which should point to an Organization.

- *descriptors relevant to the particular sample, such as*
 - sex*
 - age*
 - development stage*
 - organism part (tissue)*
 - cell type*
 - animal/plant strain or line*
 - genetic variation (e.g., gene knockout, transgenic variation)*
 - individual genetic characteristics (e.g., disease alleles, polymorphisms)*

- *disease state or normal*
- *is additional clinical information available (link)*
- *the individual (for interrelation of the samples in the experiment)*

All these are encoded as characteristics (OntologyEntries), just like the species. Additionally, if there is some crucial information about the BioSource that does not fit into OntologyEntries, a Description object can be used.

2) *Biomaterial manipulations: laboratory protocol, including relevant parameters, e.g.,*

- *growth conditions*
- *in vivo treatments (organism or individual treatments)*
- *in vitro treatments (cell culture conditions)*
- *treatment type (e.g., small molecule, heat shock, cold shock, food deprivation)*
- *compound*
- *separation technique (e.g., none, trimming, microdissection, FACS)*

For recommendations for controlled vocabularies that can be used see

<http://www.mged.org/ontology>

Manipulations are recorded with a Treatment object, which is a child object of the BioSample object, the result of this treatment (see lower right part of the object diagram). A chain of several treatments resulting in a BioSample is encoded as many Treatment objects, the order is specified by the "order" attribute. However, in simple cases just one Treatment object per BioSample is sufficient. To indicate what was treated a BioMaterialMeasurement object pointing to the source BioMaterial (in this case the BioSource object) is used. Treatment action must be provided as an OntologyEntry. A Protocol object is linked with the Treatment object via a ProtocolApplication object (Protocol is something reusable that can be used in many instance Treatments, while a ProtocolApplication corresponds to a specific Treatment). A ProtocolApplication must have an activityDate, use "n/a" if not relevant. A Person object can be attached to the ProtocolApplication if it is important to track who did what (e.g., for quality control purposes). Also, to encode protocol information in a more granular way, ParameterValues can be attached to the ProtocolApplication object, then the corresponding Protocol should have corresponding Parameter objects; see the object diagram.

BioSample also can have characteristics (OntologyEntries) and a Description, if there is important information about the BioSample that is different from the original BioSource.

3) *Hybridization extract preparation protocol for each extract prepared from the sample, including*

- *extraction method*
- *whether total RNA, mRNA, or genomic DNA is extracted*
- *amplification (RNA polymerases, PCR)*

In MAGE extracts and extract preparation recording is done in the same way as sample and sample preparation recording, since from the information structure point of view the situation is the same (see upper left part of the object diagram). There is a BioMaterial obtained as a result of a certain Treatment, which is characterized by an associated ProtocolApplication/Protocol (and optionally ParameterValues/Parameters), performed by a certain Person. In this case the source is a BioSample (pointed to by the BioMaterialMeasurement object). The type association

with an OntologyEntry "extract" should be used to indicate that this BioSample is a hybridization extract.

4) *Labeling protocol for each labeling prepared from the extract, including*

- *amount of nucleic acids labeled*
- *label used (e.g., A-Cy3, G-Cy5, 33P,)*
- *label incorporation method*

Again, the same MAGE structures are employed, only this time a LabeledExtract object must be used, and the label should be indicated by a link to the corresponding Compound object. The Compound object may be characterized by some compound index (OntologyEntry), and its name should be provided.

5) *External controls added to hybridization extract(s) (spiking controls)*

- *element on array expected to hybridize to spiking control*
- *spike type (e.g., oligonucleotide, plasmid DNA, transcript)*
- *spike qualifier (e.g., concentration, expected ratio, labelling methods if different than that of the extract)*

There are no specialized structures in MAGE for this purpose. A spike can be defined as a BioSample and described using all available above-described structures for BioMaterials. The fact of adding a spike to a sample should be encoded as a Treatment with two BioMaterialMeasurements, where one points to the main BioSample and the other points to the BioSample corresponding to the spike.

Hybridization procedures and parameters

Each hybridization description should include

1) *information about which labeled extract (related to which sample, which extract) and which array (e.g., array design, batch and serial number) has been used in the experiment;*

The Hybridization objects must have as a child object a BioMaterialMeasurement object that points to the corresponding LabeledExtract object(s). A link between the Hybridization object and the corresponding Array object also must be provided.

2) *the hybridization protocol, normally including*

- *the solution (e.g., concentration of solutes)*
- *blocking agent*
- *wash procedure*
- *quantity of labeled target used*
- *time, concentration, volume, temperature*
- *description of the hybridization instruments*

Hybridization, just like Treatment described above, in MAGE is a subclass of the BioEvent class, and ProtocolApplications can be attached to BioEvents. Here for capturing the protocol information the same classes as above (for Treatments) can be used: ProtocolApplication and Protocol (mandatory), Parameter, ParameterValue and Person (optional). Additionally, since hybridizations are always performed using some instruments, this has to be specified. Hardware objects (with attributes URI, model and make) can be used to describe instruments, and Protocol objects can point to the corresponding Hardware objects (if a certain protocol requires use of a

certain instrument). HardwareApplication objects are attached to ProtocolApplication objects and point to the corresponding Hardware objects (structure similar to ParameterValues/Parameters). If usage of Hardware includes setting certain parameters, ParameterValues and Parameters attached to the corresponding ProtocolApplications and Protocols can be used.

Measurement data and specifications of data processing

This is a separate MIAME section, but in MAGE structures tightly linked to those used for hybridization data capture are used, therefore "measurement data" part of this section is shown on the same diagrams, DTD fragment and XML example as for the previous section.

We distinguish between three levels of data processing – raw data (images), image quantitations and gene expression data matrix. Each hybridization has at least one image, each image has a corresponding image quantitation table, where a row represents an array design element and a column to a different quantitation types, such as mean or median pixel intensity. Several quantitation tables can be combined using data processing metrics to obtain the 'final' gene expression measurement table associated with the experiment.

In MAGE gene expression experiments consist of BioAssays. There are 3 types of BioAssays, PhysicalBioAssay (corresponds to all wet-lab activities plus slide scanning), MeasuredBioAssay (feature extraction) and DerivedBioAssay (data transformations). All BioAssays should point to the corresponding FactorValues, defined as children of ExperimentalFactors (see above). A Hybridization object is a child object of a PhysicalBioAssay (hybridization is bioassay's creation event).

1) Raw data description should include

- *for each scan laboratory protocol for scanning, including scanning hardware and software, scan parameters, including laser power, spatial resolution, pixel space, PMT voltage;*

ImageAcquisition is child object of a PhysicalBioAssay, and it is similar to Hybridization and Treatment objects in the sense that ProtocolApplications can be attached to it. Protocol specification is done in the same way as for Hybridizations.

- *scanned images;*
It should be noted that MGED does not have consensus whether the provision of images is a part of MIAME.

ImageAcquisition results in a new PhysicalBioAssay, and an Image can be optionally attached to this new PhysicalBioAssay. Image objects are also pointed to by the respective ImageAcquisition objects. An Image object must have a pointer to the corresponding Channel object, which in turn must reference the compound used for labeling the extract for which expression is detected in this Channel (this is how image/sample correspondence for multi-channel experiments can be traced). Image objects also have mandatory format specification (an OntologyEntry).

2) Image analysis and quantitation

- *image analysis software specification and version, availability, and the description or identification of the algorithm and all the parameters used*

A FeatureExtraction object belongs to a MeasuredBioAssay, and it points to the corresponding source PhysicalBioAssay (for tracing which Image served as a source for the particular FeatureExtraction event. FeatureExtraction is another BioEvent, therefore a ProtocolApplication can be attached to it. Protocol specification is similar to that for ImageAcquisition, except here SoftwareApplication and Software objects are used instead of HardwareApplication and Hardware. Software is defined by its name and URI. If some parameters are set in the course of using the Software, these can be attached to the ProtocolApplication and Protocol objects, similarly as for Hardware (see above).

- *for each image the complete image analysis output (of the particular image analysis software)*

A MeasuredBioAssay object points to a MeasuredBioAssayData object that holds the actual data ("image quantitation" in MIAME) that came out of the feature extraction software.

Now we will consider the "Measurement data and specifications of data processing" diagrams and XML fragments. In MAGE expression data is essentially a 3-dimensional matrix, and for image quantitation data the dimensions are BioAssayDimension, FeatureDimension and QuantitationTypeDimension. The BioAssayDimension in the simplest case contains just a single MeasuredBioAssay, the same one that points to the MeasuredBioAssayData under consideration. The FeatureDimension is an ordered list of pointers to all features that this data object corresponds to (usually each row corresponds to one Feature). The QuantitationTypeDimension points to all QuantitationTypes represented in this data matrix (usually each column corresponds to a QuantitationType).

Each QuantitationType has to be described; "scale" and "dataType" are mandatory OntologyEntries, and also a textual Description should be given. If a quantitation type corresponds to a particular Channel (e.g., cy5), it should be indicated. A Quantitation type can be further described by a set of OntologyEntries attached to its Description object. Please note that QuantitationType is an abstract class with a number of subclasses (see DTD or MAGE-OM documentation for details), and always a specific subclass is used when describing a quantitation type (e.g., MeasuredSignal), providing information on the overall nature of the quantitation type.

3) Normalized and summarized data – gene expression data matrix

- *data processing protocol, including normalization algorithm (for detailed recommendations, see <http://www.mged.org/normalization>)*

The DerivedBioAssayData object that contains the normalized and summarized data (described below) has as its child object a Transformation that points to all MeasuredBioAssayData objects that served as sources for the data normalization/ transformation. The Transformation object also contains a ProtocolApplication object that points to the Protocol object where data processing should be described; the ProtocolApplication can also point to a Person who performed the Transformation.

Transformations link BioAssayData objects, and BioAssayMaps link the corresponding BioAssays; the purpose of this dual linking in MAGE is to provide a possibility to trace data processing on higher level, involving only BioAssays, as well as on a more detailed level, involving the actual data and transformations.

- *gene expression data table(s) derived from the experiment as the whole,*
 - *derived measurement value summarizing related elements and replicates as used by the author (this may constitute replicates of the element on the same or different arrays or hybridizations, as well as different elements related to the same entity e.g., gene)*

DerivedBioAssayData is very similar to MeasuredBioAssayData described above. In this case there are several options for the DesignElementDimension - it can be either FeatureDimension, ReporterDimension, or CompositeSequenceDimension. QuantitationType dimension structurally looks the same, except there will be different QuantitationTypes used for normalized data than for measured data.

- *providing a reliability indicator for each datapoint (e.g., standard deviation) is encouraged*

There are two StandardQuantitationTypes, PValue and Error, for reporting reliability indicators. SpecializedQuantitationTypes can be used as well for this purpose.

Array design description (1)

The array design specification consists of the description of the common features of the array as the whole, and the description of each array design elements (e.g., each spot). Following terminology used in MAGE, we distinguish between three levels of array design elements: feature – the location on the array, reporter – the nucleotide sequence present in a particular location on the array, and composite sequence – a set of reporters used collectively to measure an expression of a particular gene, exon, or splice-variant. The details that should be given of each of them are described below.

PhysicalArrayDesign object should be used in MAGE to describe an array design.

1) Array related information

- *array design name*

PhysicalArrayDesign has "name" attribute.

- *platform type: in situ synthesized, spotted or other*

Since in principle the same array can contain features of several platform types, in MAGE this is defined on the FeatureGroup level. A PhysicalArrayDesigns can have more than one FeatureGroup, each of which contain similar features. A FeatureGroup must have an OntologyEntry attached via the "technologyType" association.

- *surface and coating specification*

Described by an OntologyEntry object attached to the PhysicalArrayDesign via the "surfaceType" association.

- *physical dimensions of array support (e.g. of slide)*

In MAGE this information is not attached to the PhysicalArrayDesign, or anywhere in the ArrayDesign package. Separation is made between array type description (blueprint - in ArrayDesign, DesignElement and BioSequence packages) and description of specific arrays (in the Array package). The main class in the Array package is Array, and objects of this class

correspond to individual arrays referenced from Hybridizations (see below). Each Array must point to a PhysicalArrayDesign. Many Arrays can correspond to a single ArrayGroup object, if they have been physically manufactured on a single slide. This ArrayGroup object then corresponds to "array support" in MIAME, and it has attributes length and width (of numeric type). DistanceUnit must be attached to enable interpretation of these numbers.

- *number of features on the array*

PhysicalArrayDesign has "numberOfFeatures" attribute.

- *availability (e.g., for commercial arrays) or production protocol for custom made arrays*

A PhysicalArrayDesign object can reference one or more Contacts, i.e., Persons or Organizations. At least one must be provided. For persons links to their Organizations must be provided.

To link production protocols ArrayManufacture class objects are used. An ArrayManufacture corresponds to a process of Array production, more than one Array can correspond to the same ArrayManufacture object. ArrayManufacture must have a ProtocolApplication object, which points to a Protocol object where the production protocol is described.

2a) *For each reporter type*

- *the type of the reporter: synthetic oligo-nucleotides, PCR products, plasmids, colonies, other*
- *single or double stranded*

A PhysicalArrayDesign can point to one or more ReporterGroups, used for grouping together similar Reporters. For each ReporterGroup species of its Reporters must be indicated (as an OntologyEntry), as well as other information common to all Reporters in the group (via association "types" to OntologyEntries).

2b) *For each reporter ...* - see the next section

3a) *For each feature type*

- *dimensions*

FeatureGroup class has attributes featureLength, featureWidth and featureHeight; a DistanceUnit object must be attached to indicate the unit used.

- *attachment (covalent/ionic/other)*

This should be specified, similarly as for ReporterGroups, by using OntologyEntries associated to FeatureGroups via the "types" association.

In MAGE also CompositeSequences can be grouped into CompositeSequenceGroups, and this is a way to associate CompositeSequences to PhysicalArrayDesigns. Like ReporterGroups and FeatureGroups, CompositeSequenceGroups can have more than one type.

Also belonging to the ArrayDesign package is the Zone class, used for indicating where Features are located (see below). Zones belong to ZoneGroups, which belong to PhysicalArrayDesigns. The Zone class has row and column attributes.

Array design description (2)

2b) For each reporter

- *sequence or PCR primer information:*
 - sequence or a reference sequence (e.g., for oligonucleotides), if known*

Reporters can have one or more BioSequence objects attached to define what these Reporters actually are. The BioSequence class has "sequence" attribute. Each BioSequence also must have its polymer type OntologyEntry (e.g., RNA or DNA), as well as its "type" OntologyEntry (e.g., gene or exon).

- sequence accession number in DDBJ/EMBL/GenBank, if exists*

Each BioSequence can be characterized by more than one DatabaseEntry object, which contains "accession" attribute and must point to a Database object to define in which database this accession number is valid.

- primer pair information, if relevant*

Primers are BioSequences that also characterize a Reporter, and appropriate BioSequence types must be defined by OntologyEntries.

- *approximate lengths if exact sequence not known*

The BioSequence class has "length" attribute, as well as "isApproximateLength" boolean attribute.

- *clone information, if relevant (clone ID, clone provider, date, availability)*

Also clone information can be provided using BioSequence objects, e.g., by attaching DatabaseEntries that point to the relevant clone information database. For other, less structured information (e.g., provider), a Description object attached to the Reporter can be used.

- *element generation protocol that includes sufficient information to reproduce the element for custom-made arrays that are not generally available*

This can be a part of a Description object attached to the Reporter.

3a) For each feature type ... - see the previous section

3b) For each feature

- *which reporter and the location on the array*

For mapping Reporters to Features FeatureReporterMap objects are used. These are independent, first class objects, allowing remapping if necessary. In the minimal encoding each Reporter points to one FeatureReporterMap, each of these can have many FeatureInformation objects as its children. Each of the FeatureInformation objects points to a single Feature object.

A Feature must have its row and column defined, for this purpose FeatureLocation attached to the Feature is used. Every Feature must point to its Zone (described above).

4) For each composite sequence

- *which reporters it contains*

ReporterCompositeMaps are similar to FeatureReporterMaps. Each CompositeSequence points to one ReporterCompositeMap, which can contain several ReporterPosition objects, each of them pointing to one Reporter.

- *the reference sequence*

A CompositeSequence has one BioSequence attached, and its "sequence" attribute should be used for indicating the reference sequence.

- *gene name and links to appropriate databases (e.g., SWISS-PROT, or organism specific databases), if known and relevant*

The BioSequence object can have more than one DatabaseEntry for linking to appropriate databases.

5) Control elements on the array

- *position of the feature (the abstract coordinate on the array)*
- *control type (spiking, normalization, negative, positive)*

If a Reporter or a CompositeSequence is a control, an OntologyEntry indicating its control type can be attached to it.

- *control qualifier (endogenous, exogenous)*

For each array that is not generally available (e.g., commercially available), the provided information should be sufficient to reproduce the array and all its design features.

Other MAGE structures

This page provides guidance for other auxiliary MAGE structures which are necessary for MIAME encoding. 4 MAGE packages are considered here:

- 1) QuantitationType package. Used and discussed in the "Measurement data and specifications of data processing" section.
- 2) Protocol package. These structures are used throughout MAGE. Here DTD is provided, along with 3 UML diagrams displaying structures for specifying Protocols, specifying ProtocolApplications, as well as which MAGE classes can have ProtocolApplications attached (are subclasses of the BioEvent abstract class). Also, an example MAGE-ML document providing Protocol, Hardware and Software definitions is attached.
- 3) AuditAndSecurity package. Specifies Person and Organization classes also used throughout MAGE.
- 4) Description package. Descriptions and OntologyEntries are used in all areas of MAGE, and DatabaseEntries can be referenced from BioSequences.

MAGE-ML-Lite DTD fragments

Packages are provided in the same order as required by MAGE-ML DTD.

AuditAndSecurity

```
<!--
  AuditAndSecurity_package

  Specifies classes that allow tracking of changes and information on
  user permissions to view the data and annotation.
-->
<!ELEMENT AuditAndSecurity_package (Contact_assnlist?, ..) >

<!--
  Element and Attlist declarations for Contact

  A contact is either a person or an organization.
-->
<!ELEMENT Contact_assnlist ((Person |
  Organization)+) >

<!--
  Element and Attlist declarations for Organization

  Organizations are entities like companies, universities, government
  agencies for which the attributes are self describing.
-->

<!ELEMENT Organization_ref EMPTY >
<!ATTLIST Organization_ref identifier CDATA #REQUIRED >

<!ELEMENT Organization (..) >
<!ATTLIST Organization
  identifier CDATA #REQUIRED
  name CDATA #IMPLIED
  URI CDATA #IMPLIED
  address CDATA #IMPLIED
  .. >

<!--
  Element and Attlist declarations for Person

  A person for which the attributes are self describing.

  Associations

  affiliation: The organization a person belongs to.
-->

<!ELEMENT Person_ref EMPTY >
<!ATTLIST Person_ref identifier CDATA #REQUIRED >

<!ELEMENT Person (..,Affiliation_assnref?) >
<!ATTLIST Person
  identifier CDATA #REQUIRED
  name CDATA #IMPLIED
  ..
```

```
    address CDATA #IMPLIED
    phone CDATA #IMPLIED
    ..
    email CDATA #IMPLIED
    .. >
```

```
<!ELEMENT Affiliation_assnref (Organization_ref) >
```

Description (and BQS)

```
<!--
```

```
    Description_package
```

The classes in this package allow a variety of references to third party annotation and direct annotation by the experimenter.

```
-->
```

```
<!ELEMENT Description_package (Database_assnlist?) >
```

```
<!--
```

```
    Element and Attlist declarations for Database
```

An address to a repository.

Attributes

version: The version for which a DatabaseReference applies.

URI: The location of the Database.

```
-->
```

```
<!ELEMENT Database_assnlist (Database+) >
```

```
<!ELEMENT Database_assnref (Database_ref) >
```

```
<!ELEMENT Database_ref EMPTY >
```

```
<!ATTLIST Database_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT Database (...) >
```

```
<!ATTLIST Database
```

```
    identifier CDATA #REQUIRED
```

```
    name CDATA #IMPLIED
```

```
    version CDATA #IMPLIED
```

```
    URI CDATA #IMPLIED >
```

```
<!--
```

```
    Element and Attlist declarations for DatabaseEntry
```

A reference to a record in a database.

Associations

database: Reference to the database where the DataEntry instance can be found.

Attributes

accession: The identifier used to look up the record.

accessionVersion: The appropriate version of the accession (if applicable).

URI: The location of the record.

-->

```
<!ELEMENT SequenceDatabases_assnlist (DatabaseEntry+) >
```

```
<!ELEMENT DatabaseEntry (...Database_assnref) >
```

```
<!ATTLIST DatabaseEntry
    accession CDATA #REQUIRED
    accessionVersion CDATA #IMPLIED
    URI CDATA #IMPLIED >
```

<!--

Element and Attlist declarations for Description

A free text description of an object.

Associations

annotations: Allows specification of ontology entries related to the instance being described.

bibliographicReferences: References to existing literature.

Attributes

Inherits attributes from base class Describable.

text: The description.

URI: A reference to the location and type of an outside resource.

-->

```
<!ELEMENT Descriptions_assnlist (Description+) >
```

```
<!ELEMENT QualityControlDescription_assn (Description) >
```

```
<!ELEMENT ReplicateDescription_assn (Description) >
```

```
<!ELEMENT Description (...Annotations_assnlist?,...,
    BibliographicReferences_assnlist?) >
```

```
<!ATTLIST Description
    text CDATA #IMPLIED
    URI CDATA #IMPLIED >
```

<!--

Element and Attlist declarations for OntologyEntry

A single entry from an ontology or a controlled vocabulary. For instance, category could be 'species name', value could be 'homo sapiens' and ontology would be taxonomy database, NCBI.

Associations

ontologyReference: Many ontology entries will not yet have formalized ontologies. In those cases, they will not have a database reference to the ontology.

In the future it is highly encouraged that these ontologies be

developed and ontologyEntry be subclassed from DatabaseReference.

associations: Allows an instance of an OntologyEntry to be further qualified.

Attributes

Inherits attributes from base class Extendable.

category: The category to which this entry belongs.

value: The value for this entry in this category.

description: The description of the meaning for this entry.

```
-->
<!ELEMENT Action_assn (OntologyEntry) >
<!ELEMENT Annotations_assnlist (OntologyEntry+) >
<!ELEMENT Associations_assnlist (OntologyEntry+) >
<!ELEMENT Category_assn (OntologyEntry) >
<!ELEMENT Characteristics_assnlist (OntologyEntry+) >
<!ELEMENT CompoundIndices_assnlist (OntologyEntry+) >
<!ELEMENT ControlType_assn (OntologyEntry) >
<!ELEMENT DataType_assn (OntologyEntry) >
<!ELEMENT FailTypes_assnlist (OntologyEntry+) >
<!ELEMENT Format_assn (OntologyEntry) >
<!ELEMENT MaterialType_assn (OntologyEntry) >
<!ELEMENT Parameters_assnlist (OntologyEntry+) >
<!ELEMENT PolymerType_assn (OntologyEntry) >
<!ELEMENT Scale_assn (OntologyEntry) >
<!ELEMENT Species_assn (OntologyEntry) >
<!ELEMENT SubstrateType_assn (OntologyEntry) >
<!ELEMENT SurfaceType_assn (OntologyEntry) >
<!ELEMENT TechnologyType_assn (OntologyEntry) >
<!ELEMENT Type_assn (OntologyEntry) >
<!ELEMENT Types_assnlist (OntologyEntry+) >
<!ELEMENT Value_assn (OntologyEntry) >

<!ELEMENT OntologyEntry (..,OntologyReference_assn?,
    Associations_assnlist?) >
<!ATTLIST OntologyEntry
    category CDATA #REQUIRED
    value CDATA #REQUIRED
    description CDATA #IMPLIED >

<!--
    BQS_package

    Allows a reference to an article, book or other publication to be
    specified for searching repositories.
-->

<!--
    Element and Attlist declarations for BibliographicReference

    Attributes for the most common criteria and association with
    OntologyEntry allows criteria to be specified for searching for a
    Bibliographic reference.
```

```

-->
<!ELEMENT BibliographicReferences_assnlist (BibliographicReference+) >

<!ELEMENT BibliographicReference (..,Parameters_assnlist,..) >
<!ATTLIST BibliographicReference
    title CDATA #IMPLIED
    authors CDATA #IMPLIED
    publication CDATA #IMPLIED
    publisher CDATA #IMPLIED
    editor CDATA #IMPLIED
    year CDATA #IMPLIED
    volume CDATA #IMPLIED
    issue CDATA #IMPLIED
    pages CDATA #IMPLIED
    URI CDATA #IMPLIED >

```

Protocol

```

<!--
    Protocol_package

    Provides a relatively immutable class, Protocol, that can describe a
    generic laboratory procedure or analysis algorithm, for example, and
    an instance class, ProtocolApplication, which can describe the
    actual application of a protocol. The ProtocolApplication provides
    values for the replaceable parameters of the Protocol.
-->
<!ELEMENT Protocol_package (Hardware_assnlist?,
    Software_assnlist?,
    Protocol_assnlist?) >

<!ELEMENT Hardware_assnlist (Hardware+) >
<!ELEMENT Software_assnlist (Software+) >
<!ELEMENT Protocol_assnlist (Protocol+) >

<!--
    Element and Attlist declarations for Hardware

    Hardware represents the hardware used. Examples of Hardware
    include: computers, scanners, wash stations etc...
-->

<!ELEMENT Hardware_ref EMPTY >
<!ATTLIST Hardware_ref identifier CDATA #REQUIRED >

<!ELEMENT Hardware (..) >
<!ATTLIST Hardware
    identifier CDATA #REQUIRED
    ..
    URI CDATA #IMPLIED
    model CDATA #IMPLIED
    make CDATA #IMPLIED >

<!--
    Element and Attlist declarations for Software

```

Software represents the software used. Examples of Software include: feature extraction software, clustering software, etc...

-->

```
<!ELEMENT Software_ref EMPTY >
<!ATTLIST Software_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT Software (..) >
<!ATTLIST Software
  identifier CDATA #REQUIRED
  name CDATA #IMPLIED
  URI CDATA #IMPLIED >
```

<!--

Element and Attlist declarations for Protocol

A Protocol is a parameterizable description of a method. ProtocolApplication is used to specify the ParameterValues of it's Protocol's Parameters.

Attributes

text: The text description of the Protocol.

title: The title of the Protocol

-->

```
<!ELEMENT Protocol_ref EMPTY >
<!ATTLIST Protocol_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT Protocol (..,ParameterTypes_assnlist?,
  Type_assn,
  Hardwares_assnreflist?,
  Softwares_assnreflist?) >
```

```
<!ATTLIST Protocol
  identifier CDATA #REQUIRED
  ..
  URI CDATA #IMPLIED
  text CDATA #IMPLIED
  .. >
```

```
<!ELEMENT ParameterTypes_assnlist (Parameter+) >
<!ELEMENT Hardwares_assnreflist (Hardware_ref+) >
<!ELEMENT Softwares_assnreflist (Software_ref+) >
```

<!--

Element and Attlist declarations for Parameter

A Parameter is a replaceable value in a Parameterizable class. Examples of Parameters include: scanning wavelength, laser power, centrifuge speed, multiplicative errors, the number of input nodes to a SOM, and PCR temperatures.

-->

```
<!ELEMENT Parameter_ref EMPTY >
```

```

<!ATTLIST Parameter_ref identifier CDATA #REQUIRED >

<!ELEMENT Parameter (...) >
<!ATTLIST Parameter
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED >

<!--
    Element and Attlist declarations for ProtocolApplication

    The use of a protocol with the requisite Parameters and
    ParameterValues.

    Associations

        hardwareApplications: The use of hardware for the application of
        the protocol.

        softwareApplications: The use of software for the application of
        the protocol.

        protocol: The protocol that is being used.

-->
<!ELEMENT ProtocolApplications_assnlist (ProtocolApplication+) >

<!ELEMENT ProtocolApplication (..,ParameterValues_assnlist?,
    HardwareApplications_assnlist?,
    SoftwareApplications_assnlist?,
    Performers_assnreflist?,
    Protocol_assnref) >
<!ATTLIST ProtocolApplication
    activityDate CDATA #REQUIRED >

<!ELEMENT ParameterValues_assnlist (ParameterValue+) >
<!ELEMENT HardwareApplications_assnlist (HardwareApplication+) >
<!ELEMENT SoftwareApplications_assnlist (SoftwareApplication+) >
<!ELEMENT Performers_assnreflist (Person_ref+) >
<!ELEMENT Protocol_assnref (Protocol_ref) >

<!--
    Element and Attlist declarations for ParameterValue

    The value of a Parameter.

    Associations

        parameterType: The parameter this value is for.

    Attributes

        value: The value of the parameter. Will have the datatype of
        its associated Parameter.
-->

<!ELEMENT ParameterValue (..,ParameterType_assnref) >
<!ATTLIST ParameterValue

```

```

        value CDATA #IMPLIED >

<!ELEMENT ParameterType_assnref (Parameter_ref) >

<!--
  Element and Attlist declarations for HardwareApplication

  The use of a piece of hardware with the requisite Parameters and
  ParameterValues.

  Associations

    hardware: The underlying hardware.
-->

<!ELEMENT HardwareApplication (...Hardware_assnref) >

<!ELEMENT Hardware_assnref (Hardware_ref) >

<!--
  Element and Attlist declarations for SoftwareApplication

  The use of a piece of software with the requisite Parameters and
  ParameterValues.

  Associations
    Inherits associations from base class
    ParameterizableApplication.

    software: The underlying software.

  Attributes

    version: The version of the software.
-->

<!ELEMENT SoftwareApplication (...Software_assnref) >
<!ATTLIST SoftwareApplication
  version CDATA #IMPLIED
  .. >

<!ELEMENT Software_assnref (Software_ref) >

```

BioMaterial

```

<!--
  BioMaterial_package

  The classes in this package describe how a BioSource is treated to
  obtain the BioMaterial (typically a LabeledExtract) that is used by
  a BioAssayCreation in combination with an Array to produce a
  PhysicalBioAssay. A set of treatments are typically linear in time
  but can form a Directed Acyclic Graph.
-->

```

```

<!ELEMENT BioMaterial_package (Compound_assnlist?,
    BioMaterial_assnlist?) >

<!ELEMENT Compound_assnlist (Compound+) >
<!ELEMENT BioMaterial_assnlist ((BioSource |
    LabeledExtract |
    BioSample)+) >

<!--
    Element and Attlist declarations for Compound

    A Compound can be a simple compound such as SDS (sodium dodecyl
    sulfate).

    Attributes

        isSolvent: A Compound may be a special case Solvent.
-->

<!ELEMENT Labels_assnreflist (Compound_ref+) >

<!ELEMENT Compound_ref EMPTY >
<!ATTLIST Compound_ref identifier CDATA #REQUIRED >

<!ELEMENT Compound (..,CompoundIndices_assnlist?,..) >
<!ATTLIST Compound
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    isSolvent CDATA "false" >

<!ELEMENT CompoundIndices_assnlist (OntologyEntry+) >

<!--
    Element and Attlist declarations for BioMaterial

    BioMaterial is an abstract class that represents the important
    substances such as cells, tissues, DNA, proteins, etc...
    Biomaterials can be related to other biomaterial through a directed
    acyclic graph (represented by treatment(s)).

    Associations

        characteristics: Innate properties of the biosource, such as
        genotype, cultivar, tissue type, cell type, ploidy, etc.

        materialType: The type of material used, i.e. rna, dna, lipid,
        phosphoprotein, etc.

        treatments: This association is one way from BioMaterial to
        Treatment. From this a BioMaterial can discover the amount and type
        of BioMaterial that was part of the treatment that produced it.
-->

<!--
    Element and Attlist declarations for BioSource

```

The BioSource is the original source material before any treatment events. It is also a top node of the directed acyclic graph generated by treatments. The association to OntologyEntry allows enumeration of a BioSource's inherent properties.

Associations

Inherits associations from base class BioMaterial.

sourceContact: The BioSource's source is the provider of the biological material (a cell line, strain, etc...). This could be the ATTC (American Tissue Type Collection).

Attributes

Inherits attributes from base class BioMaterial.

-->

```
<!ELEMENT BioSource_ref EMPTY >
```

```
<!ATTLIST BioSource_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT BioSource (..,Descriptions_assnlist?,...,
    Characteristics_assnlist?,
    MaterialType_assn,..,
    SourceContact_assnreflist?) >
```

```
<!ATTLIST BioSource
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED >
```

```
<!ELEMENT Characteristics_assnlist (OntologyEntry+) >
```

```
<!ELEMENT MaterialType_assn (OntologyEntry) >
```

```
<!ELEMENT SourceContact_assnreflist (Person_ref+) >
```

<!--

Element and Attlist declarations for LabeledExtract

LabeledExtracts are special BioSamples that have Compounds which are detectable (these are often fluorescent or reactive moieties).

Associations

Inherits associations from base class BioMaterial.

labels: Compound used to label the extract.

Attributes

Inherits attributes from base class BioMaterial.

-->

```
<!ELEMENT LabeledExtract_ref EMPTY >
```

```
<!ATTLIST LabeledExtract_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT LabeledExtract (..,Descriptions_assnlist?,...,
    Characteristics_assnlist?,
    MaterialType_assn,
    Treatments_assnlist?,
    Labels_assnreflist) >
```

```
<!ATTLIST LabeledExtract
    identifier CDATA #REQUIRED
    .. >
```

```
<!ELEMENT Treatments_assnlist (Treatment+) >
```

```

<!--
  Element and Attlist declarations for Treatment

  The process by which a biomaterial is created (from source
  biomaterials).  Treatments have an order and an action.

  Associations

    action: The event that occurred (e.g. grow, wait, add, etc...).
    The actions should be a recommended vocabulary

    sourceBioMaterialMeasurements: The BioMaterials and the amounts
    used in the treatment

  Attributes

    order: The chronological order in which a treatment occurred (in
    relation to other treatments).  More than one treatment can have the
    same chronological order indicating that they happened (or were
    caused to happen) simultaneously.
-->

<!ELEMENT Treatment (..,ProtocolApplications_assnlist?,
  Action_assn,..,
  SourceBioMaterialMeasurements_assnlist?) >
<!ATTLIST Treatment
  identifier CDATA #REQUIRED
  ..
  order CDATA #IMPLIED >

<!ELEMENT Action_assn (OntologyEntry) >
<!ELEMENT SourceBioMaterialMeasurements_assnlist (BioMaterialMeasurement+) >

<!--
  Element and Attlist declarations for BioMaterialMeasurement

  A BioMaterialMeasurement is a pairing of a source BioMaterial and an
  amount (Measurement) of that BioMaterial.

  Associations

    bioMaterial: A source BioMaterial for a treatment.
-->

<!ELEMENT BioMaterialMeasurement (..,BioMaterial_assnref,..) >

<!ELEMENT BioMaterial_assnref (BioSource_ref |
  LabeledExtract_ref |
  BioSample_ref) >

<!--
  Element and Attlist declarations for BioSample

  BioSamples are products of treatments that are of interest.
  BioSamples are often used as the sources for other biosamples.  The

```

Type attribute describes the role the BioSample holds in the treatment hierarchy. This type can be an extract.

Associations

Inherits associations from base class BioMaterial.

type: The Type attribute describes the role the BioSample holds in the treatment hierarchy. This type can be an extract.

Attributes

Inherits attributes from base class BioMaterial.

-->

```
<!ELEMENT BioSample_ref EMPTY >
<!ATTLIST BioSample_ref identifier CDATA #REQUIRED >

<!ELEMENT BioSample (...Descriptions_assnlist?,...,
    Characteristics_assnlist?,
    MaterialType_assn,
    Treatments_assnlist?,
    Type_assn?) >
<!ATTLIST BioSample
    identifier CDATA #REQUIRED
    .. >
```

BioSequence

<!--

BioSequence_package

Describes a known gene or sequence. BioAssays typically seek to identify what BioSequences are expressed in a BioMaterial after treatments, the expression level measured from the association between the BioMaterial and the Array. The Array's Features typically provide known locations for this association to occur. Most often, the Reporter and CompositeSequence are known and the presence or absence of a particular BioSequence in the BioMaterial is based on whether there has been an association to the DesignElement targeted for it. Some other experiments may not know the DesignElement's target but can discover it with known properties of the BioSequences in the BioMaterial.

-->

```
<!ELEMENT BioSequence_package (BioSequence_assnlist?) >
```

<!--

Element and Attlist declarations for BioSequence

A BioSequence is a representation of a DNA, RNA, or protein sequence. It can be represented by a Clone, Gene, or the sequence.

Associations

sequenceDatabases: References an entry in a sequence database, like GenBank, UniGene, etc.

polymerType: A choice of protein, RNA, or DNA.

type: The type of biosequence, i.e. gene, exon, UniGene cluster, fragment, BAC, EST, etc.

Attributes

length: The number of residues in the biosequence.

isApproximateLength: If length not positively known will be true

sequence: The actual components of the sequence, for instance, for DNA a string consisting of A,T,C and G.

The attribute is optional and instead of specified here, can be found through the DatabaseEntry.

-->

```
<!ELEMENT BioSequence_assnlist (BioSequence+) >
```

```
<!ELEMENT BioSequence_ref EMPTY >
```

```
<!ATTLIST BioSequence_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT BioSequence (...SequenceDatabases_assnlist?,...,  
    PolymerType_assn,  
    Type_assn,..) >
```

```
<!ATTLIST BioSequence
```

```
    identifier CDATA #REQUIRED
```

```
    ..
```

```
    length CDATA #IMPLIED
```

```
    isApproximateLength CDATA #IMPLIED
```

```
    ..
```

```
    sequence CDATA #IMPLIED >
```

```
<!ELEMENT SequenceDatabases_assnlist (DatabaseEntry+) >
```

```
<!ELEMENT PolymerType_assn (OntologyEntry) >
```

<!--

Element and Attlist declarations for DatabaseEntry

A reference to a record in a database.

Associations

database: Reference to the database where the DataEntry instance can be found.

Attributes

accession: The identifier used to look up the record.

accessionVersion: The appropriate version of the accession (if applicable).

URI: The location of the record.

-->

```
<!ELEMENT DatabaseEntry (...Database_assnref) >
```

```
<!ATTLIST DatabaseEntry
```

```
    accession CDATA #REQUIRED
```

```

        accessionVersion CDATA #IMPLIED
        URI CDATA #IMPLIED >

<!--ELEMENT Database_assnref (Database_ref) >

DesignElement
<!--
    DesignElement_package

    The classes of this package are the contained classes of the
    ArrayDesign and describe through the DesignElements what is intended
    to be at each location of the Array. The Feature describes an
    intended location on the Array, the Reporter the Oligo, Clone, PCR
    Product that is on a Feature and the CompositeSequence which
    combines Reporters or CompositeSequences into what the child
    DesignElements are meant to represent biologically, e.g. a Gene,
    Exon, SpliceVariant, etc.
-->
<!--ELEMENT DesignElement_package (CompositeSequence_assnlist?,
    Reporter_assnlist?,...,
    ReporterCompositeMap_assnlist?,
    FeatureReporterMap_assnlist?) >

<!--ELEMENT CompositeSequence_assnlist (CompositeSequence+) >
<!--ELEMENT Reporter_assnlist (Reporter+) >
<!--ELEMENT ReporterCompositeMap_assnlist (ReporterCompositeMap+) >
<!--ELEMENT FeatureReporterMap_assnlist (FeatureReporterMap+) >

<!--
    Element and Attlist declarations for CompositeSequence

    A collection of Reporter or CompositeSequence Design Elements,
    annotated through the association to BioSequence.

    Associations

    biologicalCharacteristics: The annotation on the BioSequence
    this CompositeSequence represents. Typically the sequences will be
    a Genes, Exons, or SpliceVariants.

    reporterCompositeMaps: A map to the reporters that compose this
    CompositeSequence.
-->

<!--ELEMENT CompositeSequences_assnreflist (CompositeSequence_ref+) >

<!--ELEMENT CompositeSequence_ref EMPTY >
<!--ATTLIST CompositeSequence_ref identifier CDATA #REQUIRED >

<!--ELEMENT CompositeSequence (...Descriptions_assnlist?,...,
    ControlType_assn?,
    BiologicalCharacteristics_assnreflist?,
    (ReporterCompositeMaps_assnreflist?|..)) >
<!--ATTLIST CompositeSequence

```

```

        identifier CDATA #REQUIRED
        name CDATA #IMPLIED >

<!ELEMENT ControlType_assn (OntologyEntry) >
<!ELEMENT BiologicalCharacteristics_assnreflist (BioSequence_ref+) >
<!ELEMENT ReporterCompositeMaps_assnreflist (ReporterCompositeMap_ref+) >

<!--
  Element and Attlist declarations for Reporter

  A Design Element that represents some biological material (clone,
  oligo, etc.) on an array which will report on some biosequence or
  biosequences.  The derived data from the measured data of its
  Features represents the presence or absence of the biosequence or
  biosequences it is reporting on in the BioAssay.

  Reporters are Identifiable and several Features on the same array
  can be mapped to the same reporter as can Features from a different
  ArrayDesign.  The granularity of the Reporters independence is
  dependent on the technology and the intent of the ArrayDesign.
  Oligos using mature technologies can in general be assumed to be
  safely replicated on many features where as with PCR Products there
  might be the desire for quality assurance to make reporters one to
  one with features and use the mappings to CompositeSequences for
  replication purposes.

  Associations

      immobilizedCharacteristics: The sequence annotation on the
      BioMaterial this reporter represents.  Typically the sequences will
      be an Oligo Sequence, Clone or PCR Primer.

      featureReporterMaps: Associates features with their reporter.

-->

<!ELEMENT Reporters_assnreflist (Reporter_ref+) >

<!ELEMENT Reporter_ref EMPTY >
<!ATTLIST Reporter_ref identifier CDATA #REQUIRED >

<!ELEMENT Reporter (.,Descriptions_assnlist?,.,.,
        ControlType_assn?,
        ImmobilizedCharacteristics_assnreflist?,.,.,
        FailTypes_assnlist?,
        FeatureReporterMaps_assnreflist?) >
<!ATTLIST Reporter
        identifier CDATA #REQUIRED
        name CDATA #IMPLIED >

<!ELEMENT ImmobilizedCharacteristics_assnreflist (BioSequence_ref+) >
<!ELEMENT FailTypes_assnlist (OntologyEntry+) >
<!ELEMENT FeatureReporterMaps_assnreflist (FeatureReporterMap_ref+) >

<!--
  Element and Attlist declarations for ReporterCompositeMap

```

A ReporterCompositeMap is the description of how source Reporters are transformed into a target CompositeSequences. For instance, several reporters that tile across a section of a chromosome could be mapped to a CompositeSequence.

Associations

compositeSequence: A map to the reporters that compose this CompositeSequence.

reporterPositionSources: Association to the reporters that compose this CompositeSequence and where those reporters occur.

-->

```
<!ELEMENT ReporterCompositeMap_ref EMPTY >
<!ATTLIST ReporterCompositeMap_ref identifier CDATA #REQUIRED >

<!ELEMENT ReporterCompositeMap (..,CompositeSequence_assnref,
    ReporterPositionSources_assnlist) >
<!ATTLIST ReporterCompositeMap
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT CompositeSequence_assnref (CompositeSequence_ref) >
<!ELEMENT ReporterPositionSources_assnlist (ReporterPosition+) >
```

<!--

Element and Attlist declarations for ReporterPosition

The location in the composite target's sequence to which a source reporter maps.

Associations

reporter: A reporter that comprises part of a CompositeSequence.

-->

```
<!ELEMENT ReporterPosition (..,Reporter_assnref,..) >
<!ELEMENT Reporter_assnref (Reporter_ref) >
```

<!--

Element and Attlist declarations for FeatureReporterMap

A FeatureReporterMap is the description of how source features are transformed into a target reporter. These would map replicate features for a reporter to the reporter.

Associations

reporter: Associates features with their reporter.

featureInformationSources: Typically, the features on an array that are manufactured with this reporter's BioSequence.

```

-->

<!ELEMENT FeatureReporterMap_ref EMPTY >
<!ATTLIST FeatureReporterMap_ref identifier CDATA #REQUIRED >

<!ELEMENT FeatureReporterMap (..,Reporter_assnref,
    FeatureInformationSources_assnlist) >
<!ATTLIST FeatureReporterMap
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT FeatureInformationSources_assnlist (FeatureInformation+) >

<!--
    Element and Attlist declarations for FeatureInformation

    A part of the map information.

    Associations

        feature: The feature the FeatureInformation is supplying
        information for.
-->

```

```

-->

<!ELEMENT FeatureInformation (..,Feature_assnref,..) >

<!ELEMENT Feature_assnref (Feature_ref) >

```

ArrayDesign

```

<!--
    ArrayDesign_package

    Describes a microarray design that can be printed and then, in the
    case of gene expression, hybridized. An array design consists of
    several features (also called spots) in which reporter sequences are
    placed. Many features may have the same reporter replicated and a
    reporter may be specified in one or more array designs.

    The nature of the reporter's biosequence placed on a spot will
    depend on the technology. Two well-known technologies differ
    significantly-spotter arrays draw material from a well and place a
    spot on the array whereas in situ oligo arrays are created through
    the synthesis of many, short (~20-100mer) nucleotide sequences onto
    the features.

    Reporters can be grouped together into CompositeSequences, typically
    representing a gene or one or more splice variants in gene
    expression experiments.

    There are then two distinct ways that DesignElements are grouped.
    The one described in the ArrayDesign package by FeatureGroup,
    ReporterGroup and CompositeGroup is by technology type, that is, one
    might want to segregate the controls to a Group and all the
    non-controls to another. Or if PCR Product and Oligos are both
    used on an array they would likely be in different groups. The
-->

```

```

    grouping described in the DesignElement package by the mappings
    relates the Features to the Reporter, the Reporters to
    CompositeSequence, and at higher levels, CompositeSequences to
    CompositeSequence.
-->
<!ELEMENT ArrayDesign_package (ReporterGroup_assnlist?,
    CompositeGroup_assnlist?,
    ArrayDesign_assnlist?) >

<!--
    Element and Attlist declarations for ReporterGroup

    Allows specification of the type of Reporter Design Element.

    Associations

        reporters: The reporters that belong to this group.
-->
<!ELEMENT ReporterGroup_assnlist (ReporterGroup+) >

<!ELEMENT ReporterGroup_ref EMPTY >
<!ATTLIST ReporterGroup_ref identifier CDATA #REQUIRED >

<!ELEMENT ReporterGroup (...Types_assnlist?,
    Species_assn?,
    Reporters_assnreflist) >
<!ATTLIST ReporterGroup
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT Species_assn (OntologyEntry) >

<!--
    Element and Attlist declarations for CompositeGroup

    Allows specification of the type of Composite Design Element.

    Associations

        compositeSequences: The compositeSequences that belong to this
        group.
-->
<!ELEMENT CompositeGroup_assnlist (CompositeGroup+) >

<!ELEMENT CompositeGroup_ref EMPTY >
<!ATTLIST CompositeGroup_ref identifier CDATA #REQUIRED >

<!ELEMENT CompositeGroup (...Types_assnlist?,
    Species_assn?,
    CompositeSequences_assnreflist) >
<!ATTLIST CompositeGroup
    identifier CDATA #REQUIRED
    .. >

```

```

<!--
  Element and Attlist declarations for ArrayDesign

  Describes the design of an gene expression layout.  In some cases
  this might be virtual and, for instance, represent the output from
  analysis software at the composite level without reporters or
  features.

  Associations
    Inherits associations from base class Identifiable.

    protocolApplications: Describes the application of any
    protocols, such as the methodology used to pick oligos, in the
    design of the array.

    featureGroups: The grouping of like Features together.
    Typically for a physical array design, this will be a single
    grouping of features whose type might be PCR Product or Oligo.  If
    more than one technology type occurs on the array, such as the
    mixing of Cloned BioMaterial and Oligos, then there would be
    multiple FeatureGroups to segregate the technology types.

    reporterGroups: The grouping of like Reporter together.  If more
    than one technology type occurs on the array, such as the mixing of
    Cloned BioMaterial and Oligos, then there would be multiple
    ReporterGroups to segregate the technology types.

    compositeGroups: The grouping of like CompositeSequence
    together.  If more than one technology type occurs on the array,
    such as the mixing of Cloned BioMaterial and Oligos, then there
    would be multiple CompositeGroups to segregate the technology types.

    designProviders: The primary contact for information on the
    array design

  Attributes

    numberOfFeatures: The number of features for this array
-->
<!ELEMENT ArrayDesign_assnlist ((..|PhysicalArrayDesign)+) >
<!--
  Element and Attlist declarations for PhysicalArrayDesign

  A design that is expected to be used to manufacture physical arrays.

  Associations
    Inherits associations from base class ArrayDesign.

    surfaceType: The type of surface from a controlled vocabulary
    that would include terms such as non-absorptive, absorptive, etc.

    zoneGroups: In the case where the array design is specified by
    one or more zones, allows specifying where those zones are located.

  Attributes

```

```

        Inherits attributes from base class ArrayDesign.
-->
<!ELEMENT PhysicalArrayDesign_ref EMPTY >
<!ATTLIST PhysicalArrayDesign_ref identifier CDATA #REQUIRED >

<!ELEMENT PhysicalArrayDesign (..,FeatureGroups_assnlist?,
    ReporterGroups_assnreflist?,
    CompositeGroups_assnreflist?,
    DesignProviders_assnreflist?,
    SurfaceType_assn?,
    ZoneGroups_assnlist?) >
<!ATTLIST PhysicalArrayDesign
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    ..
    numberOfFeatures CDATA #IMPLIED >

<!ELEMENT FeatureGroups_assnlist (FeatureGroup+) >
<!ELEMENT ReporterGroups_assnreflist (ReporterGroup_ref+) >
<!ELEMENT CompositeGroups_assnreflist (CompositeGroup_ref+) >
<!ELEMENT DesignProviders_assnreflist ((Person_ref |
    Organization_ref)+) >
<!ELEMENT SurfaceType_assn (OntologyEntry) >
<!ELEMENT ZoneGroups_assnlist (ZoneGroup+) >

<!--
    Element and Attlist declarations for FeatureGroup

    A collection of like features.

    Associations

        technologyType: The technology type of this design. By
        specifying a technology type, higher level analysis can use
        appropriate algorithms to compare the results from multiple arrays.
        The technology type may be spotted cDNA or in situ photolithography.

        distanceUnit: The unit for the feature measures.

        features: The features that belong to this group.

    Attributes

        featureWidth: The width of the feature.

        featureLength: The length of the feature.

        featureHeight: The height of the feature.
-->

<!ELEMENT FeatureGroup (..,Types_assnlist?,...,
    TechnologyType_assn?,...,
    DistanceUnit_assn?,
    Features_assnlist) >
<!ATTLIST FeatureGroup
    identifier CDATA #REQUIRED
    ..

```

```

        featureWidth CDATA #IMPLIED
        featureLength CDATA #IMPLIED
        featureHeight CDATA #IMPLIED >

<!-- TechnologyType_assn (OntologyEntry) >
<!-- Features_assnlist (Feature+) >

<!--
  Element and Attlist declarations for Feature

  An intended position on an array.

  Associations

    position: The position of the feature on the array, relative to
    the top, left corner.

    zone: A reference to the zone this feature is in.

    featureLocation: Location of this feature relative to a grid.

-->

<!-- Feature_ref EMPTY >
<!-- Feature_ref identifier CDATA #REQUIRED >

<!-- Feature (..,Zone_assnref?,
  FeatureLocation_assn?) >
<!-- Feature
  identifier CDATA #REQUIRED
  .. >

<!-- Zone_assnref (Zone_ref) >
<!-- FeatureLocation_assn (FeatureLocation) >

<!--
  Element and Attlist declarations for FeatureLocation

  Specifies where a feature is located relative to a grid.

  Attributes

    row: row position in the Zone

    column: column position in the Zone.

-->

<!-- FeatureLocation (..) >
<!-- FeatureLocation
  row CDATA #REQUIRED
  column CDATA #REQUIRED >

<!--
  Element and Attlist declarations for ZoneGroup

  Specifies a repeating area on an array. This is useful for printing

```

when the same pattern is repeated in a regular fashion.

Associations

zoneLocations: Describes the location of different zones within the array design.

-->

```
<!ELEMENT ZoneGroup (..,ZoneLocations_assnlist?) >
```

<!--

Element and Attlist declarations for Zone

Specifies the location of a zone on an array.

Attributes

row: row position in the ZoneGroup

column: column position in the ZoneGroup.

-->

```
<!ELEMENT ZoneLocations_assnlist (Zone+) >
```

```
<!ELEMENT Zone_ref EMPTY >
```

```
<!ATTLIST Zone_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT Zone (..) >
```

```
<!ATTLIST Zone
```

```
    identifier CDATA #REQUIRED
```

```
    ..
```

```
    row CDATA #IMPLIED
```

```
    column CDATA #IMPLIED
```

```
    .. >
```

Array

<!--

Array_package

Describes the process of creating arrays from array designs. Includes information on how arrays are grouped together, if relevant.

-->

```
<!ELEMENT Array_package (ArrayGroup_assnlist?,  
    Array_assnlist?,  
    ArrayManufacture_assnlist?) >
```

```
<!ELEMENT ArrayGroup_assnlist (ArrayGroup+) >
```

```
<!ELEMENT Array_assnlist (Array+) >
```

```
<!ELEMENT ArrayManufacture_assnlist (ArrayManufacture+) >
```

<!--

Element and Attlist declarations for ArrayGroup

An array package is a physical platform that contains one or more

arrays that are separately addressable (e.g. several arrays that can be hybridized on a single microscope slide) or a virtual grouping together of arrays.

The array package that has been manufactured has information about where certain artifacts about the array are located for scanning and feature extraction purposes.

Associations

arrays: Association between an ArrayGroup and its Arrays, typically the ArrayGroup will represent a slide and the Arrays will be the manufactured so that they may be hybridized separately on that slide.

distanceUnit: The unit of the measurement attributes.

Attributes

numArrays: This attribute defines the number of arrays on a chip or a slide.

width: The width of the platform

length: The length of the platform.

-->

```
<!ELEMENT ArrayGroup_ref EMPTY >
```

```
<!ATTLIST ArrayGroup_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT ArrayGroup (..,Arrays_assnreflist,  
    SubstrateType_assn?,  
    DistanceUnit_assn?) >
```

```
<!ATTLIST ArrayGroup  
    identifier CDATA #REQUIRED  
    ..  
    numArrays CDATA #IMPLIED  
    ..  
    width CDATA #IMPLIED  
    length CDATA #IMPLIED >
```

```
<!ELEMENT Arrays_assnreflist (Array_ref+) >
```

```
<!ELEMENT SubstrateType_assn (OntologyEntry) >
```

```
<!--
```

Element and Attlist declarations for Array

The physical substrate along with its features and their annotation

Associations

arrayDesign: The association of a physical array with its array design.

information: Association between the manufactured array and the information on that manufacture.

arrayGroup: Association between an ArrayGroup and its Arrays, typically the ArrayGroup will represent a slide and the Arrays will be the manufactured so that they may be hybridized separately on that slide.

-->

```
<!ELEMENT Array_ref EMPTY >
<!ATTLIST Array_ref identifier CDATA #REQUIRED >

<!ELEMENT Array (...ArrayDesign_assnref,
    Information_assnref,
    ArrayGroup_assnref?,...) >
<!ATTLIST Array
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT ArrayDesign_assnref (...|PhysicalArrayDesign_ref) >
<!ELEMENT Information_assnref (ArrayManufacture_ref) >
<!ELEMENT ArrayGroup_assnref (ArrayGroup_ref) >
```

<!--

Element and Attlist declarations for ArrayManufacture

Describes the process by which arrays are produced.

Associations

arrays: Association between the manufactured array and the information on that manufacture.

protocolApplications: The protocols followed in the manufacturing of the arrays.

-->

```
<!ELEMENT ArrayManufacture_ref EMPTY >
<!ATTLIST ArrayManufacture_ref identifier CDATA #REQUIRED >

<!ELEMENT ArrayManufacture (...Arrays_assnreflist,...,
    ProtocolApplications_assnlist?) >
<!ATTLIST ArrayManufacture
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    .. >
```

BioAssay

<!--

BioAssay_package

Provides classes that contain information and annotation on the event of joining an Array with a BioMaterial preparation, the acquisition of images and the extraction of data on a per feature basis from those images. The derived classes of BioAssay represent the base PhysicalBioAssays which lead to the production of Images, the MeasuredBioAssay which is associated with the set of

quantitations produced by FeatureExtraction, and DerivedBioAssay (see BioAssayData package) which groups together BioAssays that have been analyzed together to produce further refinement of the quantitations.

The design of this package and the related BioAssayData package was driven by the following query considerations and the desire to return as little data as necessary to satisfy a query. Often, the first set of queries for experiments below the Experiment level will want to discover the why of an experiment and this is captured in the BioAssay class through its FactorValue, BioEvent and Description associations. This separates it from the data but allows an overview of the experiment hierarchy. The BioAssayData class association to BioDataValues is optional only to allow queries on them to discover the how of the experiment through the association to the transformation and mappings of the three BioAssayData dimensions and the protocols used. Once a researcher, for instance, has narrowed down the experiments of interest then the actual data, represented by the BioDataValues, can be downloaded. Because these data can be in the hundreds of megabytes to gigabytes range, it was considered desirable to be able to return information and annotation on the experiment without the data.

-->

```
<!ELEMENT BioAssay_package (Channel_assnlist?,  
    BioAssay_assnlist?) >
```

```
<!ELEMENT Channel_assnlist (Channel+) >
```

```
<!ELEMENT BioAssay_assnlist ((PhysicalBioAssay |  
    DerivedBioAssay |  
    MeasuredBioAssay)+) >
```

<!--

Element and Attlist declarations for Channel

A channel represents an independent acquisition scheme for the ImageAcquisition event, typically a wavelength.

Associations

labels: The compound used to label the extract.

-->

```
<!ELEMENT Channel_ref EMPTY >
```

```
<!ATTLIST Channel_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT Channel (..,Labels_assnreflist?) >
```

```
<!ATTLIST Channel  
    identifier CDATA #REQUIRED  
    name CDATA #IMPLIED >
```

<!--

Element and Attlist declarations for BioAssay

An abstract class which represents both physical and computational groupings of arrays and biomaterials.

Associations

bioAssayFactorValues: The values that this BioAssay is associated with for the experiment.

-->

```
<!ELEMENT BioAssays_assnreflist ((PhysicalBioAssay_ref |
    DerivedBioAssay_ref |
    MeasuredBioAssay_ref)+) >
```

<!--

Element and Attlist declarations for PhysicalBioAssay

A bioAssay created by the bioAssayCreation event (e.g. in gene expression analysis this event is represented by the hybridization event).

Associations

Inherits associations from base class BioAssay.

physicalBioAssayData: The Images associated with this PhysicalBioAssay by ImageAcquisition.

bioAssayCreation: The association between the BioAssayCreation event (typically Hybridization) and the PhysicalBioAssay and its annotation of this event.

bioAssayTreatments: The set of treatments undergone by this PhysicalBioAssay.

Attributes

Inherits attributes from base class BioAssay.

-->

```
<!ELEMENT PhysicalBioAssay_ref EMPTY >
<!ATTLIST PhysicalBioAssay_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT PhysicalBioAssay (...Channels_assnreflist?,
    BioAssayFactorValues_assnreflist?,
    PhysicalBioAssayData_assnlist?,
    BioAssayCreation_assn?,
    BioAssayTreatments_assnlist?) >
```

```
<!ATTLIST PhysicalBioAssay
    identifier CDATA #REQUIRED
    .. >
```

```
<!ELEMENT Channels_assnreflist (Channel_ref+) >
<!ELEMENT BioAssayFactorValues_assnreflist (FactorValue_ref+) >
<!ELEMENT PhysicalBioAssayData_assnlist (Image+) >
<!ELEMENT BioAssayCreation_assn (Hybridization) >
<!ELEMENT BioAssayTreatments_assnlist ((...|ImageAcquisition)+) >
```

<!--

Element and Attlist declarations for Image

An image is created by an imageAcquisition event, typically by

scanning the hybridized array (the PhysicalBioAssay).

Associations

format: The file format of the image typically a TIF or a JPEG.

Attributes

URI: The file location in which an image may be found.

-->

```
<!ELEMENT Image_ref EMPTY >
<!ATTLIST Image_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT Image (..,Channels_assnreflist?,
                Format_assn) >
<!ATTLIST Image
    identifier CDATA #REQUIRED
    ..
    URI CDATA #IMPLIED >
```

```
<!ELEMENT Format_assn (OntologyEntry) >
```

```
<!--
```

Element and Attlist declarations for Hybridization

The archetypal bioAssayCreation event, whereby biomaterials are hybridized to an array.

```
-->
```

```
<!ELEMENT Hybridization (..,ProtocolApplications_assnlist?,
                        SourceBioMaterialMeasurements_assnlist?,
                        Array_assnref) >
<!ATTLIST Hybridization
    identifier CDATA #REQUIRED
    .. >
```

```
<!ELEMENT Array_assnref (Array_ref) >
```

```
<!--
```

Element and Attlist declarations for BioAssayTreatment

The event which records the process by which PhysicalBioAssays are processed.

Associations

physicalBioAssay: The set of treatments undergone by this PhysicalBioAssay.

target: The PhysicalBioAssay that was treated.

```
-->
```

```
<!--
```

Element and Attlist declarations for ImageAcquisition

The process by which an image is generated (typically scanning).

Associations

images: The images produced by the ImageAcquisition event.

-->

```
<!ELEMENT ImageAcquisition (..,ProtocolApplications_assnlist?,  
    Target_assnref,  
    Images_assnreflist?) >
```

```
<!ATTLIST ImageAcquisition  
    identifier CDATA #REQUIRED  
    .. >
```

```
<!ELEMENT Target_assnref (PhysicalBioAssay_ref) >
```

```
<!ELEMENT Images_assnreflist (Image_ref+) >
```

<!--

Element and Attlist declarations for DerivedBioAssay

A BioAssay that is created by the Transformation BioEvent from one or more MeasuredBioAssays or DerivedBioAssays.

Associations

Inherits associations from base class BioAssay.

derivedBioAssayData: The data associated with the DerivedBioAssay.

derivedBioAssayMap: The DerivedBioAssay that is produced by the sources of the BioAssayMap.

Attributes

Inherits attributes from base class BioAssay.

-->

```
<!ELEMENT DerivedBioAssay_ref EMPTY >
```

```
<!ATTLIST DerivedBioAssay_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT DerivedBioAssay (..,BioAssayFactorValues_assnreflist?,...,  
    DerivedBioAssayData_assnreflist?,  
    DerivedBioAssayMap_assnreflist?) >
```

```
<!ATTLIST DerivedBioAssay  
    identifier CDATA #REQUIRED  
    .. >
```

```
<!ELEMENT DerivedBioAssayData_assnreflist (DerivedBioAssayData_ref+) >
```

```
<!ELEMENT DerivedBioAssayMap_assnreflist (BioAssayMap_ref+) >
```

<!--

Element and Attlist declarations for MeasuredBioAssay

A measured bioAssay is the direct processing of information in a physical bioAssay by the featureExtraction event. Often uses images which are referenced through the physical bioAssay.

Associations

Inherits associations from base class BioAssay.

featureExtraction: The association between the MeasuredBioAssay and the FeatureExtraction Event.

measuredBioAssayData: The data associated with the MeasuredBioAssay.

Attributes

Inherits attributes from base class BioAssay.

-->

```
<!ELEMENT MeasuredBioAssay_ref EMPTY >
<!ATTLIST MeasuredBioAssay_ref identifier CDATA #REQUIRED >

<!ELEMENT MeasuredBioAssay (...,BioAssayFactorValues_assnreflist?,
    FeatureExtraction_assn?,
    MeasuredBioAssayData_assnreflist?) >
<!ATTLIST MeasuredBioAssay
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT FeatureExtraction_assn (FeatureExtraction) >
<!ELEMENT MeasuredBioAssayData_assnreflist (MeasuredBioAssayData_ref+) >
```

<!--

Element and Attlist declarations for FeatureExtraction

The process by which data is extracted from an image producing a measuredBioAssayData and a measuredBioAssay.

Associations

physicalBioAssaySource: The PhysicalBioAssay used in the FeatureExtraction event.

-->

```
<!ELEMENT FeatureExtraction (...,ProtocolApplications_assnlist?,
    PhysicalBioAssaySource_assnref) >
<!ATTLIST FeatureExtraction
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT PhysicalBioAssaySource_assnref (PhysicalBioAssay_ref) >
```

QuantitationType

<!--

QuantitationType_package

This Package defines the classes for quantitations, such as measured and derived signal, error, and pvalue. The subclasses of StandardQuantitationType will be the best fit from FeatureExtraction or Transformation Protocol for the values obtained. Other values can be specified using SpecializedQuantitationType.

```

-->
<!ELEMENT QuantitationType_package (QuantitationType_assnlist?) >

<!--
  Element and Attlist declarations for QuantitationType

  A method for calculating a single datum of the matrix (e.g. raw
  intensity, background, error).

  Associations

      channel: The optional channel associated with the
      QuantitationType.

      scale: Indication of how to interpret the value. From a
      suggested vocabulary of {LINEAR | LN | LOG2 | LOG10 | FOLD_CHANGE |
      OTHER}

      dataType: The specific type for the quantitations. From a
      controlled vocabulary of {float, int, boolean, etc.}

  Attributes

      isBackground: Indicates whether the quantitation has been
      measured from the background or from the feature itself.
-->
<!ELEMENT QuantitationType_assnlist ((SpecializedQuantitationType |
    DerivedSignal |
    MeasuredSignal |
    Error |
    PValue |
    ExpectedValue |
    Ratio |
    PresentAbsent |
    Failed)+) >

<!--
  Element and Attlist declarations for SpecializedQuantitationType

  User defined quantitation type.

-->
<!ELEMENT SpecializedQuantitationType_ref EMPTY >
<!ATTLIST SpecializedQuantitationType_ref identifier CDATA #REQUIRED >

<!ELEMENT SpecializedQuantitationType (...Descriptions_assnlist?,...,
    Channel_assnref?,
    Scale_assn,
    DataType_assn,...) >
<!ATTLIST SpecializedQuantitationType
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    isBackground CDATA #REQUIRED >

<!ELEMENT Channel_assnref (Channel_ref) >
<!ELEMENT Scale_assn (OntologyEntry) >

```

```

<!ELEMENT DataType_assn (OntologyEntry) >

<!--
  Element and Attlist declarations for DerivedSignal

  A calculated measurement of the intensity of a signal, for example,
  after a transformation involving normalization and/or replicate
  DesignElements.  Of type float.

-->
<!ELEMENT DerivedSignal_ref EMPTY >
<!ATTLIST DerivedSignal_ref identifier CDATA #REQUIRED >

<!ELEMENT DerivedSignal (..,Descriptions_assnlist?,
  Channel_assnref?,
  Scale_assn,
  DataType_assn,..) >
<!ATTLIST DerivedSignal
  identifier CDATA #REQUIRED
  name CDATA #IMPLIED
  isBackground CDATA #REQUIRED >

<!--
  Element and Attlist declarations for Error

  Error measurement of a quantitation.  Of type float.

-->
<!ELEMENT Error_ref EMPTY >
<!ATTLIST Error_ref identifier CDATA #REQUIRED >

<!ELEMENT Error (..,Descriptions_assnlist?,...,
  Channel_assnref?,
  Scale_assn,
  DataType_assn,...,
  TargetQuantitationType_assnref) >
<!ATTLIST Error
  identifier CDATA #REQUIRED
  name CDATA #IMPLIED
  isBackground CDATA #REQUIRED >

<!ELEMENT TargetQuantitationType_assnref (SpecializedQuantitationType_ref |
  DerivedSignal_ref |
  MeasuredSignal_ref |
  Error_ref |
  PValue_ref |
  ExpectedValue_ref |
  Ratio_ref |
  PresentAbsent_ref |
  Failed_ref) >

<!--
  Element and Attlist declarations for ExpectedValue

  Indication of what value is expected of the associated standard
  quantitation type.

```

```

-->
<!ELEMENT ExpectedValue_ref EMPTY >
<!ATTLIST ExpectedValue_ref identifier CDATA #REQUIRED >

<!ELEMENT ExpectedValue (...Descriptions_assnlist?,...,
    Channel_assnref?,
    Scale_assn,
    DataType_assn,...,
    TargetQuantitationType_assnref) >
<!ATTLIST ExpectedValue
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    isBackground CDATA #REQUIRED >

<!--
    Element and Attlist declarations for Failed

    Values associated with this QuantitationType indicate a failure of
    some kind for a particular DesignElement for a BioAssay. Of type
    boolean.

-->
<!ELEMENT Failed_ref EMPTY >
<!ATTLIST Failed_ref identifier CDATA #REQUIRED >

<!ELEMENT Failed (...Descriptions_assnlist?,...,
    Channel_assnref?,
    Scale_assn,
    DataType_assn,...) >
<!ATTLIST Failed
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    isBackground CDATA #REQUIRED >

<!--
    Element and Attlist declarations for MeasuredSignal

    Best measure from feature extraction as to the presence and
    intensity of the signal. Of type float.

-->
<!ELEMENT MeasuredSignal_ref EMPTY >
<!ATTLIST MeasuredSignal_ref identifier CDATA #REQUIRED >

<!ELEMENT MeasuredSignal (...Descriptions_assnlist?,...,
    Channel_assnref?,
    Scale_assn,
    DataType_assn,...) >
<!ATTLIST MeasuredSignal
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    isBackground CDATA #REQUIRED >

<!--
    Element and Attlist declarations for PValue

    Measurement of the accuracy of a quantitation. Of type float.

```

```

-->
<!ELEMENT PValue_ref EMPTY >
<!ATTLIST PValue_ref identifier CDATA #REQUIRED >

<!ELEMENT PValue (..,Descriptions_assnlist?,...,
    Channel_assnref?,
    Scale_assn,
    DataType_assn,...,
    TargetQuantitationType_assnref) >
<!ATTLIST PValue
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    isBackground CDATA #REQUIRED >

<!--
    Element and Attlist declarations for PresentAbsent

    Indicates relative presence or absence. From the enumeration
    AbsoluteCallTypeEnum {Present | Absent | Marginal | No call} or
    ComparisonCallTypeEnum {Increase I Marginal Increase | Decrease |
    Marginal Decrease | No change | No Call | Unknown }, as specified
    by the dataType.

-->
<!ELEMENT PresentAbsent_ref EMPTY >
<!ATTLIST PresentAbsent_ref identifier CDATA #REQUIRED >

<!ELEMENT PresentAbsent (..,Descriptions_assnlist?,...,
    Channel_assnref?,
    Scale_assn,
    DataType_assn,..) >
<!ATTLIST PresentAbsent
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    isBackground CDATA #REQUIRED >

<!--
    Element and Attlist declarations for Ratio

    The ratio of two or more signals, typically between two channels.
    Of type float.

-->
<!ELEMENT Ratio_ref EMPTY >
<!ATTLIST Ratio_ref identifier CDATA #REQUIRED >

<!ELEMENT Ratio (..,Descriptions_assnlist?,...,
    Channel_assnref?,
    Scale_assn,
    DataType_assn,..) >
<!ATTLIST Ratio
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED
    isBackground CDATA #REQUIRED >

```

BioAssayData

```
<!--
  BioAssayData_package

  The classes defined here provide data and the information and
  annotation on the derivation of that data.  Some of the scenarios
  that might occur are the following.

  FeatureExtraction of a single PhysicalBioAssay produces
  MeasuredBioAssayData that has a single BioAssay on the
  BioAssayDimension, typically the Features described in the
  ArrayDesign on the DesignElementDimension, and the Quantitations
  associated with the application of a FeatureExtraction protocol on
  the QuantitationDimension.

  An error model transformation might be applied that doesn't change
  the BioAssayDimension or the DesignElementDimension but likely
  changes the QuantitationDimension.  A transformation on replicate
  Reporters or CompositeSequences might be applied on the single
  BioAssay that would change the DesignElementDimension and the
  QuantitationDimension both.  Replicate and Control BioAssays might
  be added to the BioAssayDimension and a transformation could change
  the BioAssayDimension and the QuantitationDimension but not change
  the DesignElementDimension to produce a new DerivedBioAssayData.  Or
  some combination of the above transformations could be performed at
  once to change all three dimensions.

-->
<!ELEMENT BioAssayData_package (BioAssayDimension_assnlist?,
    DesignElementDimension_assnlist?,
    QuantitationTypeDimension_assnlist?,
    BioAssayMap_assnlist?,...,
    BioAssayData_assnlist?) >

<!ELEMENT BioAssayDimension_assnlist (BioAssayDimension+) >
<!ELEMENT DesignElementDimension_assnlist ((CompositeSequenceDimension |
    ReporterDimension |
    FeatureDimension)+) >
<!ELEMENT QuantitationTypeDimension_assnlist (QuantitationTypeDimension+) >
<!ELEMENT BioAssayMap_assnlist (BioAssayMap+) >
<!ELEMENT BioAssayData_assnlist ((DerivedBioAssayData |
    MeasuredBioAssayData)+) >

<!--
  Element and Attlist declarations for BioAssayDimension

  An ordered list of bioAssays.

  Associations

    bioAssays: The BioAssays for this Dimension

-->
<!ELEMENT BioAssayDimension_ref EMPTY >
<!ATTLIST BioAssayDimension_ref identifier CDATA #REQUIRED >
```

```

<!ELEMENT BioAssayDimension (..,BioAssays_assnreflist?) >
<!ATTLIST BioAssayDimension
    identifier CDATA #REQUIRED
    .. >

<!--
    Element and Attlist declarations for DesignElementDimension

    An ordered list of designElements. It will be realized as one of its
    three subclasses.

-->
<!--
    Element and Attlist declarations for CompositeSequenceDimension

    Specialized DesignElementDimension to hold CompositeSequences.

    Associations

        compositeSequences: The CompositeSequences for this Dimension.

-->
<!ELEMENT CompositeSequenceDimension_ref EMPTY >
<!ATTLIST CompositeSequenceDimension_ref identifier CDATA #REQUIRED >

<!ELEMENT CompositeSequenceDimension (..,CompositeSequences_assnreflist?) >
<!ATTLIST CompositeSequenceDimension
    identifier CDATA #REQUIRED
    .. >

<!--
    Element and Attlist declarations for ReporterDimension

    Specialized DesignElementDimension to hold Reporters.

    Associations

        reporters: The reporters for this dimension.

-->
<!ELEMENT ReporterDimension_ref EMPTY >
<!ATTLIST ReporterDimension_ref identifier CDATA #REQUIRED >

<!ELEMENT ReporterDimension (..,Reporters_assnreflist?) >
<!ATTLIST ReporterDimension
    identifier CDATA #REQUIRED
    .. >

<!--
    Element and Attlist declarations for FeatureDimension

    Specialized DesignElementDimension to hold Features.

    Associations

        containedFeatures: The features for this dimension.

```

```

-->
<!ELEMENT FeatureDimension_ref EMPTY >
<!ATTLIST FeatureDimension_ref identifier CDATA #REQUIRED >

<!ELEMENT FeatureDimension (..,ContainedFeatures_assnreflist?) >
<!ATTLIST FeatureDimension
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT ContainedFeatures_assnreflist (Feature_ref+) >

<!--
    Element and Attlist declarations for QuantitationTypeDimension

    An ordered list of quantitationTypes.

    Associations

        quantitationTypes: The QuantitationTypes for this Dimension.
-->

<!ELEMENT QuantitationTypeDimension_ref EMPTY >
<!ATTLIST QuantitationTypeDimension_ref identifier CDATA #REQUIRED >

<!ELEMENT QuantitationTypeDimension (..,QuantitationTypes_assnreflist?) >
<!ATTLIST QuantitationTypeDimension
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT QuantitationTypes_assnreflist ((SpecializedQuantitationType_ref |
    DerivedSignal_ref |
    MeasuredSignal_ref |
    Error_ref |
    PValue_ref |
    ExpectedValue_ref |
    Ratio_ref |
    PresentAbsent_ref |
    Failed_ref)+) >

<!--
    Element and Attlist declarations for BioAssayMap

    The BioAssayMap is the description of how source MeasuredBioAssays
    and/or DerivedBioAssays are manipulated (mathematically) to produce
    DerivedBioAssays.

    Associations

        bioAssayMapTarget: The DerivedBioAssay that is produced by the
        sources of the BioAssayMap.

        sourceBioAssays: The sources of the BioAssayMap that are used to
        produce a target DerivedBioAssay.

```

```

-->

<!ELEMENT BioAssayMap_ref EMPTY >
<!ATTLIST BioAssayMap_ref identifier CDATA #REQUIRED >

<!ELEMENT BioAssayMap (..,BioAssayMapTarget_assnref,
    SourceBioAssays_assnreflist?) >
<!ATTLIST BioAssayMap
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT BioAssayMapTarget_assnref (DerivedBioAssay_ref) >
<!ELEMENT SourceBioAssays_assnreflist ((..|MeasuredBioAssay_ref)+) >

<!--
    Element and Attlist declarations for BioAssayData

    Represents the dataset created when the BioAssays are created.
    BioAssayData is the entry point to the values. Because the actual
    values are represented by a different object, BioDataValues, which
    can be memory intensive, the annotation of the transformation can be
    gotten separate from the data.

    Associations

        bioAssayDimension: The BioAssays of the BioAssayData.

        designElementDimension: The DesignElements of the BioAssayData.

        quantitationTypeDimension: The QuantitationTypes of the
        BioAssayData.

        bioDataValues: The data values of the BioAssayData.
-->
<!--
    Element and Attlist declarations for DerivedBioAssayData

    The output of a transformation event.

    Associations

        Inherits associations from base class BioAssayData.

        producerTransformation: The association between the
        DerivedBioAssayData and the Transformation event that produced it.

    Attributes

        Inherits attributes from base class BioAssayData.
-->

<!ELEMENT DerivedBioAssayData_ref EMPTY >
<!ATTLIST DerivedBioAssayData_ref identifier CDATA #REQUIRED >

<!ELEMENT DerivedBioAssayData (..,BioAssayDimension_assnref?,
    DesignElementDimension_assnref?,
    QuantitationTypeDimension_assnref?,
    BioDataValues_assn?,

```

```

        ProducerTransformation_assn?) >
<!ATTLIST DerivedBioAssayData
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT BioAssayDimension_assnref (BioAssayDimension_ref) >
<!ELEMENT DesignElementDimension_assnref (CompositeSequenceDimension_ref |
    ReporterDimension_ref |
    FeatureDimension_ref) >
<!ELEMENT QuantitationTypeDimension_assnref (QuantitationTypeDimension_ref) >
<!ELEMENT BioDataValues_assn (BioDataCube | ..) >
<!ELEMENT ProducerTransformation_assn (Transformation) >

<!--
    Element and Attlist declarations for BioDataValues

    The actual values for the BioAssayCube.

-->
<!--
    Element and Attlist declarations for BioDataCube

    A three-dimensional cube representation of the data.

    Associations

        dataInternal: Transformed class to associate white spaced
        delimited data to the BioAssayDataCube

        dataExternal: Transformed class to associate external data to
        the BioAssayDataCube

    Attributes

        order: The order to expect the dimension. The enumeration uses
        the first letter of the three dimensions to represent the six
        possible orderings.
-->
<!ELEMENT BioDataCube (...,(DataInternal_assn | DataExternal_assn)) >
<!ATTLIST BioDataCube
    order (BDQ|BQD|DBQ|DQB|QBD|QDB) "BDQ" >

<!ELEMENT DataInternal_assn (DataInternal) >
<!ELEMENT DataExternal_assn (DataExternal) >

<!--
    Element and Attlist declarations for DataInternal

    Transformed class to associate whitespaced delimited data to the
    BioAssayDataCube

    Associations
        #PCDATA: The tab delimited data.
-->

<!ELEMENT DataInternal (#PCDATA) >

```

```

<!--
  Element and Attlist declarations for DataExternal

  Transformed class to associate external data to the BioAssayDataCube

  Attributes
    dataFormat: The format of the external file, whitespace
    delimited, tab delimited, netcdf, etc...

    filenameURI: The name and location of the file containing the
    external data
-->

<!ELEMENT DataExternal EMPTY >
<!ATTLIST DataExternal dataFormat CDATA "whitespace"
    ..
    filenameURI CDATA #REQUIRED >

<!--
  Element and Attlist declarations for Transformation

  The process by which derivedBioAssays are created from
  measuredBioAssays and/or derivedBioAssays. It uses mappings to
  indicate the input and output dimensions.

  Associations

    bioAssayDataSources: The BioAssayData sources that the
    Transformation event uses to produce the target DerivedBioAssayData.
-->

<!ELEMENT Transformation (..,ProtocolApplications_assnlist?,
    BioAssayDataSources_assnreflist?,..) >
<!ATTLIST Transformation
    identifier CDATA #REQUIRED
    .. >

<!ELEMENT BioAssayDataSources_assnreflist ((.. |
    MeasuredBioAssayData_ref)+) >

<!--
  Element and Attlist declarations for MeasuredBioAssayData

  The data associated with the MeasuredBioAssay produced by
  FeatureExtraction.

  Associations
    Inherits associations from base class BioAssayData.

  Attributes
    Inherits attributes from base class BioAssayData.
-->

<!ELEMENT MeasuredBioAssayData_ref EMPTY >
<!ATTLIST MeasuredBioAssayData_ref identifier CDATA #REQUIRED >

```

```

<!ELEMENT MeasuredBioAssayData (..,BioAssayDimension_assnref?,
    DesignElementDimension_assnref?,
    QuantitationTypeDimension_assnref?,
    BioDataValues_assn?) >
<!ATTLIST MeasuredBioAssayData
    identifier CDATA #REQUIRED
    .. >

```

Experiment (and Measurement)

```

<!--
    Experiment_package

    Represents the container for a hierarchical grouping of BioAssays.
    Can have,
    through the ExperimentDesign, a description and annotation of the
    overall design of the experiment and what it was to show.
-->
<!ELEMENT Experiment_package (Experiment_assnlist?) >

<!--
    Element and Attlist declarations for Experiment

    The Experiment is the collection of all the BioAssays that are
    related by the ExperimentDesign.

    Associations

        providers: The providers of the Experiment, its data and
        annotation.

        bioAssayData: The collection of BioAssayDatas for this
        Experiment.

        bioAssays: The collection of BioAssays for this Experiment.

        experimentDesigns: The association to the description and
        annotation of the Experiment, along with the grouping of the
        top-level BioAssays.

-->
<!ELEMENT Experiment_assnlist (Experiment+) >

<!ELEMENT Experiment (..,Descriptions_assnlist?,...,
    Providers_assnreflist?,...,
    BioAssayData_assnreflist?,
    BioAssays_assnreflist?,
    ExperimentDesigns_assnlist) >
<!ATTLIST Experiment
    identifier CDATA #REQUIRED
    name CDATA #IMPLIED >

<!ELEMENT Providers_assnreflist (Person_ref+) >
<!ELEMENT BioAssayData_assnreflist ((DerivedBioAssayData_ref |
    MeasuredBioAssayData_ref)+) >
<!ELEMENT ExperimentDesigns_assnlist (ExperimentDesign+) >

```

```

<!--
  Element and Attlist declarations for ExperimentDesign

  The ExperimentDesign is the description and collection of
  ExperimentalFactors and the hierarchy of BioAssays to which they
  pertain.

  Associations

      types: Classification of an experiment.  For example 'normal vs.
      diseased', 'treated vs. untreated', 'time course', 'tiling', etc.

      experimentalFactors: The description of the factors (TimeCourse,
      Dosage, etc.) that group the BioAssays.

      qualityControlDescription: Description of the quality control
      aspects of the Experiment.

-->

<!ELEMENT ExperimentDesign (..,Types_assnlist?,...,
      ExperimentalFactors_assnlist?,
      QualityControlDescription_assn?,...,
      ReplicateDescription_assn?) >

<!ELEMENT ExperimentalFactors_assnlist (ExperimentalFactor+) >
<!ELEMENT QualityControlDescription_assn (Description) >
<!ELEMENT ReplicateDescription_assn (Description) >

<!--
  Element and Attlist declarations for ExperimentalFactor

  ExperimentFactors are the dependent variables of an experiment (e.g.
  time, glucose concentration, ...).

  Associations

      category: The category of an ExperimentalFactor could be
      biological (time, [glucose]) or a methodological factor (differing
      cDNA preparation protocols).

      factorValues: The pairing of BioAssay FactorValues with the
      ExperimentDesign ExperimentFactor.

-->

<!ELEMENT ExperimentalFactor (..,Category_assn?,
      FactorValues_assnlist?,...) >
<!ATTLIST ExperimentalFactor
      identifier CDATA #REQUIRED
      .. >

<!ELEMENT Category_assn (OntologyEntry) >
<!ELEMENT FactorValues_assnlist (FactorValue+) >

<!--

```

Element and Attlist declarations for FactorValue

The value for a ExperimentalFactor

Associations

Inherits associations from base class Identifiable.

measurement: The measured value for this factor.

value: Allows a more complex value to be specified for a FactorValue than a simple Measurement.

-->

```
<!ELEMENT FactorValue_ref EMPTY >
```

```
<!ATTLIST FactorValue_ref identifier CDATA #REQUIRED >
```

```
<!ELEMENT FactorValue (..,(Measurement_assn? | Value_assn?)) >
```

```
<!ATTLIST FactorValue
```

```
    identifier CDATA #REQUIRED
```

```
    .. >
```

```
<!ELEMENT Measurement_assn (Measurement) >
```

```
<!ELEMENT Value_assn (OntologyEntry) >
```

```
<!--
```

Measurement_package

The classes of this package provide utility information on the quantities of other classes to each other.

-->

```
<!--
```

Element and Attlist declarations for Measurement

A Measurement is a quantity with a unit.

Associations

unit: The Unit associated with the Measurement.

Attributes

Inherits attributes from base class Extendable.

type: The type of measurement, for instance if the measurement is five feet, it can be either absolute (five feet tall) or change (five feet further along).

value: The value of the measurement. kindCV (and otherKind) determine with Unit the datatype of value.

kindCV: One of the enumeration values to determine the controlled vocabulary of the value.

otherKind: Name of the controlled vocabulary if it isn't one of the Unit subclasses.

-->

```

<!ELEMENT Measurement (Unit_assn?) >
<!ATTLIST Measurement
    type (absolute|
        change) "absolute"
    value CDATA #IMPLIED
    kindCV (time|
        distance|
        temperature|
        quantity|
        mass|
        volume|
        concentration|
        other) "other"
    otherKind CDATA #IMPLIED >

<!--
    Element and Attlist declarations for Unit

    The unit is a strict enumeration of types.

    Attributes

        unitName: The name of the unit.
-->
<!ELEMENT Unit_assn
(TimeUnit|DistanceUnit|TemperatureUnit|QuantityUnit|MassUnit|VolumeUnit|Conce
ntrationUnit) >

<!ELEMENT TimeUnit (..) >
<!ATTLIST TimeUnit
    unitName CDATA #IMPLIED
    unitNameCV (years|months|weeks|d|h|m|s|us|other) #REQUIRED >

<!ELEMENT DistanceUnit_assn (DistanceUnit) >

<!ELEMENT DistanceUnit (..) >
<!ATTLIST DistanceUnit
    unitName CDATA #IMPLIED
    unitNameCV (fm|pm|nm|um|mm|cm|m|other) #REQUIRED >

<!ELEMENT TemperatureUnit (..) >
<!ATTLIST TemperatureUnit
    unitName CDATA #IMPLIED
    unitNameCV (degree_C|degree_F|K) #REQUIRED >

<!ELEMENT QuantityUnit (..) >
<!ATTLIST QuantityUnit
    unitName CDATA #IMPLIED
    unitNameCV (mol|amol|fmol|pmol|nmol|umol|mmol|molecules|other)
#REQUIRED >

<!ELEMENT MassUnit (..) >
<!ATTLIST MassUnit
    unitName CDATA #IMPLIED
    unitNameCV (kg|g|mg|ug|ng|pg|fg|other) #REQUIRED >

```

```
<!ELEMENT VolumeUnit (..) >
<!ATTLIST VolumeUnit
    unitName CDATA #IMPLIED
    unitNameCV (mL|cc|dL|L|uL|nL|pL|fL|other) #REQUIRED >

<!ELEMENT ConcentrationUnit (..) >
<!ATTLIST ConcentrationUnit
    unitName CDATA #IMPLIED
    unitNameCV (M|mM|uM|nM|pM|fM|mg_per_mL|mL_per_L|g_per_L|
        gram_percent|mass_per_volume_percent|mass_per_mass_percent|other)
    #REQUIRED >
```