PyUCIS Documentation

Release 0.0.1

Matthew Ballance and Contributors

Jul 17, 2022

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

1.1 What is PyUCIS?

The Accellera Unified Coverage Interoperability Standard (UCIS) specifies a data model, C API, and XML interchange format for coverage metrics data. The PyUCIS library provides two APIs for creating and accessing coverage data via the UCIS data model:

- An object-oriented Python API
- A functional C-style Python API that is identical to the API defined in the Accellera standard

The PyUCIS library currently supports three back-ends for storing and accessing coverage data:

- In-Memory: An in-memory transient data model
- XML: Ability to write and read a UCIS data model to the Accellera-defined interchange format
- Library: Ability to call a tool-provided implementation of the UCIS C API

Here is a short example of using the object-oriented Python API to create a covergroup with a single coverpoint. Note that the database handle (*db*) must be obtained from the appropriate back-end factory:

```
testnode = db.createHistoryNode(
    None,
    "logicalName",
    ucisdb,
    UCIS_HISTORYNODE_TEST)
td = TestData(
    teststatus=UCIS_TESTSTATUS_OK,
    toolcategory="UCIS:simulator",
    date="20200202020"
    )
testnode.setTestData(td)

file = db.createFileHandle("dummy", os.getcwd())

srcinfo = SourceInfo(file, 0, 0)
du = db.createScope(
    "foo.bar",
    srcinfo,
    1, # weight
    UCIS_OTHER,
    UCIS_DU_MODULE,
    UCIS_ENABLED_STMT | UCIS_ENABLED_BRANCH
```

(continues on next page)

(continued from previous page)

```
    | UCIS_ENABLED_COND | UCIS_ENABLED_EXPR
    | UCIS_ENABLED_FSM | UCIS_ENABLED_TOGGLE
    | UCIS_INST_ONCE | UCIS_SCOPE_UNDER_DU
    )

instance = db.createInstance(
    "dummy",
    None, # sourceinfo
    1, # weight
    UCIS_OTHER,
    UCIS_INSTANCE,
    du,
    UCIS_INST_ONCE)

cg = instance.createCovergroup(
    "cg",
    SourceInfo(file, 3, 0),
    1, # weight
    UCIS_OTHER)

cp = cg.createCoverpoint(
    "t",
    SourceInfo(file, 4, 0),
    1, # weight
    UCIS_VLOG
    )
cp.setComment("Hello There")

cp.createBin(
    "auto[a]",
    SourceInfo(file, 4, 0),
    1,
    4,
    "a")

db.write(ucisdb, None, True, -1)
db.close()
```

1.2 Contributors

Ballance

CHAPTER

TWO

COMMANDS

The *ucis* root command accepts one of several sub-commands that operate on coverage data.

Manipulate UCIS coverage data

```
usage: ucis [-h] {convert,merge,list-db-formats,list-rpt-formats,report} ...
```

2.1 Sub-commands:

2.1.1 convert

Converts coverage data from one format to another

```
ucis convert [-h] --out OUT [--input-format INPUT_FORMAT]
             [--output-format OUTPUT_FORMAT]
             input
```

Positional Arguments

input Source database to convert

Named Arguments

--out, -o Specifies the output of the conversion

--input-format, -if Specifies the format of the input database. Defaults to 'xml'

--output-format, -of Specifies the format of the output database. Defaults to 'xml'

2.1.2 merge

Merges coverage data from two or more databases into a single merged database

```
ucis merge [-h] --out OUT [--input-format INPUT_FORMAT]
           [--output-format OUTPUT_FORMAT] [--libucis LIBUCIS]
           db [db ...]
```

Positional Arguments

db

Named Arguments

--out, -o	Specifies the output of the merge
--input-format, -if	Specifies the format of the input databases. Defaults to 'xml'
--output-format, -of	Specifies the format of the input databases. Defaults to 'xml'
--libucis, -l	Specifies the name/path of the UCIS shared library

2.1.3 list-db-formats

Shows available database formats

```
ucis list-db-formats [-h]
```

2.1.4 list-rpt-formats

Shows available report filters

```
ucis list-rpt-formats [-h]
```

2.1.5 report

Generate a report (typically textual) from coverage data

```
ucis report [-h] [--out OUT] [--input-format INPUT_FORMAT]
            [--output-format OUTPUT_FORMAT]
            db
```

Positional Arguments

db	Path to the coverage database

Named Arguments

--out, -o	Specifies the output location for the report
--input-format, -if	Specifies the format of the input database. Defaults to 'xml'
--output-format, -of	Specifies the output format of the report. Defaults to 'txt'

CHAPTER

THREE

PYUCIS REFERENCE

This section provides information on APIs and file formats used by PyUCIS.

3.1 Coverage Report Object

Many of the tools that format and visualize coverage data make use of the coverage report object implemented by PyUCIS. The *CoverageReport* object contains information about the covergroups, covergroup instances, coverpoints and cross coverpoints in a coverage database. It contains data on the percentage of coverage achieved, as well as detailed information on the number of hits in each coverpoint and cross bin.

3.1.1 Building a CoverageReport

The best way to obtain a coverage report is to use the *CoverageReportBuilder* class. The *build* method on this class accepts a UCIS database object and returns a *CoverageReport* class.

class ucis.report.**CoverageReportBuilder**(*db*)

Builds a coverage-report object from a UCIS database

static build(*db: UCIS*) → *CoverageReport*

Builds a CoverageReport object from a UCIS database

3.1.2 CoverageReport Object

The *CoverageReport* object is a tree of covergroups and coverpoints.

class ucis.report.**CoverageReport**

Root coverage-report object

coverage

Coverage percentage achieved by all covergroups

covergroups

List of (type) covergroups

class ucis.report.CoverageReport.**Covergroup**(*name*, *instname*)

Contains coverage data for a covergroup type or instance

covergroup_m: Dict[str, *Covergroup*]

Map of covergroup instance names to object This is only populated when *self* is a type covergroup

covergroups: List[*Covergroup*]

List of covergroup sub-instances. This is only populated when *self* is a type covergroup

coverpoint_m: Dict[str, *Coverpoint*]

Map of coverpoint name to object

coverpoints: List[*Coverpoint*]

List of coverpoints in the covergroup

cross_m: Dict[str, *Cross*]

Map of cross name to object

crosses: List[*Cross*]

List of cross points in the covergroup

instname

Covergroup instance name

class ucis.report.CoverageReport.**CoverItem**(*name*)

Base type for covergroups and coverpoints

coverage

Coverage percentage achieved by the cover item

name

Name of the cover item

weight

Weight given to this item when calculating coverage %

class ucis.report.CoverageReport.**Coverpoint**(*name*)

Bases: *CoverItem*

Contains coverage data about a coverpoint

coverage

Coverage percentage achieved by the cover item

name

Name of the cover item

weight

Weight given to this item when calculating coverage %

class ucis.report.CoverageReport.**Cross**(*name*)

Bases: *CoverItem*

Contains coverage data for a cross

coverage

Coverage percentage achieved by the cover item

name

Name of the cover item

weight

Weight given to this item when calculating coverage %

3.2 JSON Coverage Report

The JSON coverage-report format is provided to simplify the task of post-processing a PyUCIS coverage report. The JSON format is easily read and processed by Python as well as JSON libraries for other languages.

The JSON coverage-report format is a direct transcription of the Coverage Report Object API.

3.2.1 PyUCIS JSON Coverage Report

https://fvutils.github.io/pyucis/covreport.json			
Validation schema for the PyUCIS machine-readable coverage report			
type	*object*		
properties			
• covreport	*Coverage Report*		
	Root of the coverage report		
	type	*object*	
	properties		
	• covergroups	List of (type) covergroups	
		type	*array*
		items	*Covergroup Type*
	• coverage	Coverage percentage achieved by all covergroups	
		type	*number*
additionalProperties	False		

3.2.2 Covergroup Type

Contains information about a type covergroup		
properties		
• name	Type name of the covergroup	
	type	*string*
• coverage	Coverage percentage achieved by this covergroup type	
	type	*number*
• coverpoints	List of coverpoints	
	type	*array*
	items	*Coverpoint*
• covergroups	List of instance covergroups	
	type	*array*
	items	*Covergroup Inst*

3.2.3 Covergroup Inst

Contains information about an instance covergroup		
type	*object*	
properties		
• name	Instance name of this covergroup	
	type	*string*
• coverage	Coverage percentage achieved by this covergroup type	
	type	*number*
• coverpoints	List of coverpoints	
	type	*array*
	items	*Coverpoint*
• crosses	List of cross coverpoints	
	type	*array*
	items	*Cross*
• covergroups	List of instance covergroups	
	type	*array*
	items	*Covergroup Inst*

3.2.4 Coverpoint

Coverage information about a coverpoint		
type	*object*	
properties		
• name	Leaf name of the coverpoint	
	type	*string*
• coverage	Coverage achieved by this cross	
	type	*number*
• bins	List of coverage bins	
	type	*array*
	items	*Coverage Bin*
• ignorebins	List of ignored coverage bins	
	type	*array*
	items	*Coverage Bin*
• illegalbins	List of illegal coverage bins	
	type	*array*
	items	*Coverage Bin*

3.2.5 Cross

Coverage information about a cross coverpoint		
type	*object*	
properties		
• name	Leaf name of the cross	
	type	*string*
• coverage	Coverage achieved by this cross	
	type	*number*
• bins	List of coverage bins	
	type	*array*
	items	*Coverage Bin*

3.2.6 Cover Bin

Coverpoint or cross bin		
type	*object*	
properties		
• name	Name of the bin	
	type	*string*
• goal	Number of bin hits required to claim coverage	
	type	*integer*
• count	Number of hits the bin has	
	type	*integer*

3.3 XML Interchange Format

The Accelera UCIS Standard document specifies an XML interchange format. While the XML document structure has some similarities with the data model accessed via the UCIS C API, there are also significant differences.

The UCIS standards document also has relatively few examples of the XML interchange format, leaving some things a bit ambiguous. None of the ambiguities are with respect to the document schema. Rather, they are with respect to how a schema-compliant XML document is interpreted.

This section of the PyUCIS documentation describes how PyUCIS interprets a schema-compliant XML description, and the data it produces. Many of the details are shaped by how existing tools interpret the XML interchange format. Because our goal is to maximize interoperability, PyUCIS deliberately shapes its data output (especially) to maximize interoperability

3.3.1 Functional Coverage

Functional coverage data is stored in *cgInstance* sections within a *covergroupCoverage* scope.

Covergroup instance/type linkage

The cgId section inside the cgInstance specifies the associated covergroup type.

```
<cgInstance name="top.cg_i1" key="0">
  <options/>
  <cgId cgName="my_covergroup" moduleName="top">
    <cginstSourceId file="1" line="1" inlineCount="1"/>
    <cgSourceId file="1" line="1" inlineCount="1"/>
  </cgId>
</cgInstance>
```

This covergroup instance name is *top.cg_i1*, and is associated with a covergroup type *top::my_covergroup*.

Covergroup instance and type data

The UCIS data model represents covergroup type coverage (the merge of all covergroup instances of a given type) as a scope that contains a series of sub-scopes that hold per-instance coverage data. The XML interchange format does not provide such a hierarchy.

When instance coverage is being recorded, all cgInstance sections associated with a given covergroup type contain instance data. The reader of XML data is responsible for reconstructing type coverage.

When instance coverage is not being recorded, only a single cgInstance section is written. This section contains type data. This intepretation is backed up bythe spec: a *covergroupCoverage* scope with a single *cgInstance* entry represents coverage for the covergroup as a whole. For example, a covergroup with *per_instance* set to false.

Coverage Instance Options

When reading XML, PyUCIS considers the following coverage options significant:

- _per_instance_ - Indicates whether per-instance data is recorded
- _merge_instances_ - Specifies whether to produce type coverage from the merge of instance data

Both of these options are boolean options. PyUCIS accepts *true*, *false*, *0*, and *1*.

When writing XML, PyUCIS emits *auto_bin_max=0*. This is because PyUCIS represents all coverpoint bins explicitly. Some consumers of XML interchange format attempt to create auto-bins if this option is not explicitly set to 0.

Coverpoint Bins

```
<coverpoint name="cp1" key="0">
    <options/>
    <coverpointBin name="a[0]", type="bins" key="0">
        <range from "-1" to "-1">
            contents coverageCount="1"/>
        </range>
    </coverpointBin>
</coverpoint>
```

Coverpoint data is stored within a *coverpoint* subsection inside *cgInstance*. PyUCIS writes the bin type as one of *bins*, *ignore*, *illegal*.

PyUCIS only interprets and records the following options: - *weight* - *at_least*

PyUCIS does not interpret the value-range data, and records both bounds of the range as *-1*. This is because UCIS doesn't provide relevant data to record.

Cross Bins

The most common case with cross bins is to record auto-bins resulting from the cross of the relevant coverpoints.

```
<cross name="cp1Xcp2" key="0">
    <options/>
    <crossExpr>cp1</crossExpr>
    <crossExpr>cp2</crossExpr>
    <crossBin name="&lt;a[0],a[0]&gt;" key="0">
        <index>0</index>
        <index>0</index>
        <contents coverageCount="1"/>
    </crossBin>
</cross>
```

Note that the bin-index information is not something that is present in the UCIS data model. Cross bins, like all other bins, are simply named counts. PyUCIS attempts to reconstruct the indices by looking for bin names within the bin name. In the example above, the bin names a[0], a[0] are both the first bin within their respective coverpoints. Consequently, bin indices 0,0 are specified.

In the case of *ignore* or *illegal* bins, all indices are specified as -1.

PyUCIS records the bin type only if it is *ignore* or *illegal*. This improves interoperability with some tools.

3.4 YAML Coverage Data Format

The YAML coverage-data format is used to represent functional coverage data in a manner that is accurate and relatively easy for humans and tools to create and process.

3.4.1 Format Reference

Coverage Data

https://fvutils.github.io/pyucis/coverage.json			
PyUCIS JSON Coverage Data			
type	*object*		
properties			
• coverage	type	*object*	
	properties		
	• covergroups	List of covergroup types	
		type	*array*
		items	*Type Covergroup*

Every coverage-data document has a *coverage* element as its root. Currently, the only sub-elements is a list of covergroup types.

Type Covergroup

Holds data about a single covergroup type		
type	*object*	
properties		
• **name**	Type name of the covergroup	
	type	*string*
• weight	Weight this covergroup is given against the others	
	type	*integer*
• instances	List of covergroup instances of this type	
	type	*array*
	items	*Inst Covergroup*

A type covergroup provides data about a covergroup type. All instances of a covergroup type have the same coverpoints and crosses. All coverpoints in instances of a covergroup type have the same bins. Merged type coverage (the union of coverage achieved by all instances) is derived by PyUCIS from the instance coverage, and is not specified in the coverage file.

Inst Covergroup

Holds data about a single covergroup instance		
type	*object*	
properties		
• **name**	Instance name of this covergroup	
	type	*string*
• coverpoints	List of coverpoints	
	type	*array*
	items	*Coverpoint*
• crosses	List of crosses	
	type	*array*
	items	*Cross*

An instance covergroup provides data about a covergroup instance.

Coverpoint

Holds data about a single instance coverpoint		
type	*object*	
properties		
• **name**	Name of the coverpoint	
	type	*string*
• atleast	Number of bin hits required for coverage (default=1)	
	type	*integer*
• **bins**	List of coverage bins	
	type	*array*
	items	*Coverage Bin*
• ignorebins	List of ignore bins	
	type	*array*
	items	*Coverage Bin*
• illegalbins	List of illegal bins	
	type	*array*
	items	*Coverage Bin*

A coverpoint lists a set of bins that it is monitoring. Each coverpoint can specify an *atleast* count to specify that a bin must contain *atleast* hits in order to count as being covered. By default, *atleast* is 1.

Cross

type	*object*		
properties			
• **name**	Cross name		
	type	*string*	
• atleast	Number of bin hits required for coverage (default=1)		
	type	*integer*	
• **coverpoints**	List of coverpoint members of this cross		
	type	*array*	
	items	type	*string*
• bins	List of cross bins		
	type	*array*	
	items	*Coverage Bin*	

A cross lists the set of coverpoints from which it is composed, and lists its cross bins. Each cross can specify an *atleast* count to specify that a bin must contain *atleast* hits in order to count as being covered. By default, *atleast* is 1.

Coverage Bin

type	*object*	
properties		
• **name**	Bin name	
	type	*string*
• **count**	Hits in this bin	
	type	*integer*

A coverbin associates a bin name with the number of hits in that bin.

3.5 Best Practices for Recording Coverage

This section offers some suggestions on best practices for recording coverage using the UCIS API.

3.5.1 Naming Cross Bins

Most of the recorded cross bins will pertain to auto-cross bins between the coverpoints. Bins have arbirary names, but downstream tools (expecially the XML interchange format export) depend on being able to determine the relationship between a cross bin and its associated coverpoint bins.

The suggested name format for a cross bin is:

```
<{cp[0].bin},{cp[1].bin},...>
```

In other words, if the first bin of the first coverpoint is named 'a' and the first bin of the second coverpoint is named 'b', the cross bin for these two bins will be named: <a,b>

3.6 UCIS Object-Oriented API

Placeholder for documenting the object-oriented API

class ucis.**Obj**

getIntProperty(*coverindex: int*, *property: IntProperty*) → int

setIntProperty(*coverindex: int*, *property: IntProperty*, *value: int*)

getRealProperty(*coverindex: int*, *property: RealProperty*) → float

setRealProperty(*coverindex: int*, *property: RealProperty*, *value: float*)

getStringProperty(*coverindex: int*, *property: StrProperty*) → str

setStringProperty(*coverindex: int*, *property: StrProperty*, *value: str*)

getHandleProperty(*coverindex: int*, *property: HandleProperty*) → Scope

setHandleProperty(*coverindex: int*, *property: HandleProperty*, *value: Scope*)

accept(*v*)

3.7 UCIS C-Style API

Placeholder for documenting the C-Style UCIS API implemented by PyUCIS

ucis.__init__.**ucis_GetIntProperty**(*db: UCIS*, *obj:* Obj, *coverindex: int*, *property: IntProperty*) → int

ucis.__init__.**ucis_SetIntProperty**(*db: UCIS*, *obj:* Obj, *coverindex: int*, *property: IntProperty*, *value: int*)

ucis.__init__.**ucis_GetRealProperty**(*db: UCIS*, *obj: Obj*, *coverindex: int*, *property: RealProperty*) → float

ucis.__init__.**ucis_SetRealProperty**(*db: UCIS*, *obj: Obj*, *coverindex: int*, *property: RealProperty*, *value: float*)

ucis.__init__.**ucis_GetStringProperty**(*db: UCIS*, *obj: Obj*, *coverindex: int*, *property: StrProperty*) → str

ucis.__init__.**ucis_SetStringProperty**(*db: UCIS*, *obj: Obj*, *coverindex: int*, *property: StrProperty*, *value: str*)

ucis.__init__.**ucis_GetHandleProperty**(*db: UCIS*, *obj: Obj*, *coverindex: int*, *property: HandleProperty*) → Scope

ucis.__init__.**ucis_SetHandleProperty**(*db: UCIS*, *obj: Obj*, *coverindex: int*, *property: HandleProperty*, *value: Scope*)

ucis.__init__.**ucis_CreateScope**(*db: UCIS*, *parent: Scope*, *name: str*, *sourceinfo: SourceInfo*, *weight: int*, *source: SourceT*, *type: ScopeTypeT*, *flags*) → Scope

ucis.__init__.**ucis_CreateInstance**(*db: UCIS*, *parent: Scope*, *name: str*, *fileinfo: SourceInfo*, *weight: int*, *source: SourceT*, *type: ScopeTypeT*, *du_scope: Scope*, *flags: FlagsT*) → Scope

ucis.__init__.**ucis_CreateToggle**(*db: UCIS*, *parent: Scope*, *name: str*, *canonical_name: str*, *flags: FlagsT*, *toggle_metric: ToggleMetricT*, *toggle_type: ToggleTypeT*, *toggle_dir: ToggleDirT*) → Scope

ucis.__init__.**ucis_CreateNextCover**(*db: UCIS*, *parent: Scope*, *name: str*, *data: CoverData*, *sourceinfo: SourceInfo*) → int

ucis.__init__.**ucis_RemoveScope**(*db: UCIS*, *scope: Scope*) → int

ucis.__init__.**ucis_CreateHistoryNode**(*db: UCIS*, *parent: HistoryNode*, *logicalname*, *physicalname*, *kind: HistoryNodeKind*)

ucis.__init__.**ucis_CreateFileHandle**(*db: UCIS*, *filename: str*, *fileworkdir: str*)

ucis.__init__.**ucis_SetTestData**(*db: UCIS*, *testhistorynode: HistoryNode*, *testdata: TestData*)

ucis.__init__.**ucis_Write**(*db: UCIS*, *file: str*, *scope: Scope*, *recurse: int*, *covertype: CoverTypeT*) → int

ucis.__init__.**ucis_Close**(*db: UCIS*) → int

ucis.__init__.**ucis_Time**()

Current time in UCIS Y/M/D/H/M/S format

3.8 Back-End APIs

3.8.1 In-Memory

3.8.2 UCIS C API

3.8.3 XML

CHAPTER

FOUR

INDICES AND TABLES

PYTHON MODULE INDEX

u

INDEX

W