ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО/МЭК 19770-2— 2014

Информационные технологии

МЕНЕДЖМЕНТ ПРОГРАММНЫХ АКТИВОВ

Часть 2

Тег идентификации программного обеспечения

ISO/IEC 19770-2:2009
Information technology – Software asset management – Part 2. Software identification tag

(IDT)

Издание официальное

Москва
Стандартинформ
2015

gost.e-ecolog.ru

Предисловие

1 ПОДГОТОВЛЕН Закрытым акционерным обществом «Консистент Софтвеа Дистрибьюшн» на основе аутентичного перевода на русский язык международного стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 076 «Системы менеджмента»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 19 ноября 2014 г. № 1684-ст

4 Настоящий стандарт идентичен международному стандарту ИСО/МЭК 19770-2:2009 «Информационные технологии. Менеджмент программных активов. Часть 2. Тег идентификации программного обеспечения» (ISO/IEC 19770-2:2009) «Information technology – Software asset management – Part 2: Software identification tag»)

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты Российской Федерации, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

Правила применения настоящего стандарта установлены в ГОСТ Р 1.0–2012 (раздел 8). Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок – в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомления и тексты размещаются также в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.gost.ru)

© Стандартинформ, 2015

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область действия ... 1
1.1 Назначение ... 1
1.2 Область применения ... 1
1.3 Ограничения ... 2
2 Соответствие ... 2
2.1 Общие положения ... 2
2.2 Соответствие продуктов ... 2
2.3 Организационное соответствие ... 6
2.4 Соответствие соглашения ... 7
3 Нормативные ссылки ... 7
4 Термины, определения и сокращения ... 7
4.1 Термины и определения ... 7
4.2 Сокращения ... 11
5 Согласование и координация с предыдущими стандартами ... 12
5.1 Заявление о согласовании с данной частью настоящего стандарта ... 12
5.2 Согласование с ИСО/МЭК 19770-1:2006 Информационная технология. Управление программными активамию Часть 1. Процессы ... 12
5.3 Согласование с ИСО/МЭК 20000-1:2005 Информационная технология. Управление услугами. Часть 1. Спецификации ... 12
5.4 Согласование с ИСО/МЭК 20000-2:2005 Информационная технология. Управление услугами. Часть 2. Нормы и правила ... 13
6 Реализация процессов создания и ведения тегов идентификации программного обеспечения ... 14
6.1 Общие требования и инструкции ... 14
6.2 Жизненный цикл работы с тегами идентификации программного обеспечения: порядок операций ... 22
7 Требования и инструкции по использованию платформ ... 24
7.1 Типы платформ ... 24
7.2 Базовые сервисы платформы ... 25
7.3 Виртуальные среды ... 25
7.4 Виртуальные машины ... 25
7.5 Поддержка программного обеспечения, установленного на съемном носителе ... 26
7.6 Идентификация оборудования и платформы ... 26
8 Элементы ... 27
8.1 Общие положения ... 27
8.2 Имена элементов ... 28
8.3 Обязательные элементы ... 28
8.4 Дополнительные элементы ... 32
8.5 Расширенные элементы ... 54
8.6 Определения типов данных ... 54
Приложение А (справочное) Принципы создания и ведения тегов идентификации программного обеспечения ... 60
Приложение В (справочное) Сценарии действий и инструкции для поставщиков программного обеспечения ... 65

Приложение C (справочное) Сценарии действий и инструкции для поставщиков инструментария 70
Приложение D (справочное) Сценарии действий и инструкции для потребителей программного обеспечения 73
Приложение E (справочное) Теги идентификации программного обеспечения для элементов, не являющихся программным обеспечением 77
Приложение F (справочное) Авторское право и теги идентификации программного обеспечения 78
Приложение G (справочное) Определение схемы XML (XSD) 79
Приложение H (справочное) Расширенные примеры 87
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов национальным стандартам Российской Федерации и действующим в этом качестве межгосударственным стандартам 91
Библиография 91

Введение

Данная часть настоящего стандарта представляет собой Международный стандарт, описывающий теги идентификации программного обеспечения. Тег идентификации программного обеспечения представляет собой XML-файл, содержащий достоверную идентификационную и управляющую информацию о программном продукте. Тег идентификации программного обеспечения устанавливается на вычислительное устройство вместе с программным продуктом. Тег может создаваться в процессе установки или добавляться позднее для уже установленного программного обеспечения без тегов. Однако чаще происходит так, что тег создается при исходной разработке программного продукта, а затем распространяется и устанавливается вместе с программным продуктом. Наличие тега, начиная с самого начала разработки программного продукта, позволяет обеспечить более эффективное управление процессами распространения и переупаковки, выполняемых лицами, внешними по отношению к потребителю программного обеспечения, и последующими процессами управления релизами в организации потребителя программного обеспечения.

Данная часть настоящего стандарта поддерживает процессы управления программными активами, определенные в ИСО/МЭК 19770-1. Данный стандарт также предназначен для совместного использования с будущим ИСО/МЭК 19770-3, который будет представлять собой Международный стандарт, описывающий теги прав на использование программного обеспечения.

Теги идентификации программного обеспечения – полезный инструмент для всех заинтересованных лиц, занятых в процессах создания, лицензирования, распространения, выпуска релизов, установки и текущего технического обслуживания программного обеспечения. К основным преимуществам, связанным с использованием тегов идентификации программного обеспечения, относятся следующие:

a) возможность единообразной и достоверной идентификации программных продуктов для управления ими в любых целях, например в целях лицензирования, обновления, упаковки или для составления ведомости взаимозависимостей. Теги идентификации программного обеспечения содержат метаданные, необходимые для обеспечения более точной идентификации. Данный подход коренным образом отличается традиционных способов идентификации, ориентированных на файлы;

b) способность идентифицировать группы или наборы программных продуктов аналогично тому, как осуществляется идентификация отдельных программных продуктов, позволяет осуществлять управление целыми группами или наборами программных продуктов так же гибко, как и управление отдельными продуктами;

c) упрощение практических процессов стандартизации между различными создателями программного обеспечения и внутри самих организаций создателей программного обеспечения, определяющих, как именно следует идентифицировать различные версии программного обеспечения, что позволит потребителям программного обеспечения осуществлять более качественную идентификацию и управление этими различными версиями; например они смогут проводить различия между самостоятельными версиями и версиями, являющимися компонентами наборов, обновлений и пр.;

d) упрощение автоматизированных подходов к проверкам лицензионного соответствия с использованием информации, содержащейся как в теге идентификации программного обеспечения, так и в теге прав на использование программного обеспечения, который будет определен в ИСО/МЭК 19770-3;

e) возможность предоставления исчерпывающей информации о структурных составляющих пакетов, т. е. списка компонентов, таких как файлы и системные настройки, связанные с этим пакетом, с целью сопоставления процессов управления на уровне пакетов и на уровне файлов;

f) возможность предоставления информации о том, как именно должна осуществляться идентификация в случае активного или неактивного использования конкретного программного пакета;

g) Возможность управления сложностью программного обеспечения, установленного на съемных носителях, или носителях общего доступа, или в виртуальных средах (при условии, что платформы и программы-установщики будут совершенствовать технические возможности по идентификации устройств и окружений);

h) Возможность отражать в теге идентификации программного обеспечения идентификационные данные и требования различных объектов, в том числе создателей программного обеспечения, лицензиаров программного обеспечения, упаковщиков, дистрибьюторов, внешних по отношению к потребителю программного обеспечения, администраторов релизов в организации потребителя программного обеспечения и лиц, ответственных за установку и управление программным обеспечением, на постоянной основе;

i) Возможность выполнения проверки правильности (валидации) любой такой информации посредством дополнительного использования цифровых подписей любым лицом, создающим или изменяющим информацию в теге идентификации программного обеспечения;

j) Возможность для лиц, не являющихся создателями программного обеспечения (например, независимых поставщиков или внутреннего персонала), создавать теги идентификации программного обеспечения для устаревшего программного обеспечения, а также для программного обеспечения, поступившего от создателей программного обеспечения, которые сами не предоставили теги идентификации программного обеспечения;

k) по мере того, как отраслевые организации начнут применять единые подходы к работе с различными типами информации, не охватываемой в настоящее время данной частью настоящего стандарта, в настоящий стандарт будут вносится формальные и неформальные усовершенствующие изменения, например в части активации продуктов.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационные технологии

МЕНЕДЖМЕНТ ПРОГРАММНЫХ АКТИВОВ
ЧАСТЬ 2

Тег идентификации программного обеспечения

Information technologies. Software asset management. Part 2. Software identification tag

Дата введения – 2016–01–01

1 Область действия

1.1 Назначение

Настоящий стандарт устанавливает спецификации для создания и ведения тегов программного обеспечения с целью оптимизации процессов идентификации и управления таким программным обеспечением.

1.2 Область применения

Настоящий стандарт применяется к следующим объектам:

a) провайдеры платформы: Провайдерами платформы являются лица, отвечающие за компьютерное оборудование, или аппаратное устройство, и/или соответствующую операционную систему, или виртуальную среду, на которых может устанавливаться или запускаться программное обеспечение. Провайдеры платформы, поддерживающие данную часть настоящего стандарта, кроме того, реализуют функциональность управления тегами на уровне платформы или операционной системы;

b) поставщики программного обеспечения: Поставщиками программного обеспечения являются лица, которые создают («создатели программного обеспечения»), изменяют («модификаторы программного обеспечения») или лицензируют («лицензиары программного обеспечения») программное обеспечение для распространения или установки К ним относятся производители программного обеспечения, независимые разработчики программного обеспечения, консультанты и переупаковщики ранее произведенного программного обеспечения. Поставщиками программного обеспечения также могут считаться разработчики программного обеспечения для внутренних целей организации;

c) поставщики тегов: Поставщиками тегов являются лица, которые создают («создатели тегов») или изменяют («модификаторы тегов») теги идентификации программного обеспечения. Поставщик тегов может являться частью организации поставщика программного обеспечения или может быть сторонней организацией или потребителем программного обеспечения;

d) поставщики инструментария для работы с тегами: К ним относятся лица, которые могут предоставлять любое количество инструментальных средств, с помощью которых можно создавать, изменять или использовать теги идентификации программного обеспечения. К такому инструментарию могут относиться среды разработки, обеспечивающие автоматическое формирование тегов идентификации программного обеспечения, средства установки, которые могут создавать и/или изменять теги от имени процесса установки, а также инструменты управления настольными системами, которые могут создавать теги для программного обеспечения, не имеющего тегов, и/или изменять теги, внося в них сведения о релизе, в течение жизненного цикла программного обеспечения. Более подробная информация о возможных способах использования тегов идентификации программного обеспечения поставщиками инструментария для работы с тегами приведена в приложении С;

e) потребители программного обеспечения: Потребителями программного обеспечения являются лица, которые приобретают, устанавливают и/или иным способом потребляют программное обеспечение, и которые считаются одними из главных получателей усовершенствованной информации, содержащейся в теге идентификации программного обеспечения, как это определено в данной части настоящего стандарта. Более подробная информация о возможных способах использования тегов идентификации программного обеспечения потребителями программного обеспечения приведена в приложении D.

Издание официальное

1.3 Ограничения

В данной части настоящего стандарта не приводится описание процессов SAM, используемых для выверки прав на использование программного обеспечения с помощью тегов идентификации программного обеспечения.

В данной части настоящего стандарта не приводится описание процессов активации продуктов или контроля запуска.

Данная часть настоящего стандарта не имеет целью вступление в противоречие ни с политиками, процедурами и стандартами организации, ни с любыми внутригосударственными законами и регуляторными актами. Любое такое противоречие должно быть устранено до начала использования данной части настоящего стандарта.

2 Соответствие

2.1 Общие положения

Критерии соответствия могут применяться к продукту или организации. Для организационного соответствия определенная область действия должна охватывать как организационную область действия, так и продукты, включенные в эту область действия.

Если в отношении продукта или организации выдвигается претензия о соответствии, в претензии должна быть указана область действия, в которой было осуществлено тестирование соответствия.

В контексте данного пункта соответствие часто определяется в терминах, соответствующих требованиям 6.1, 8.3, 8.4 и 8.5. Требования к соответствию платформы также определены в 7.2. Также существуют нормативные требования, определенные в других подпунктах пунктов 6 и 7, при описании которых используется слова «должен», «должна», «должны» и пр., однако эти требования не включаются в содержание заявлений о соответствии, за исключением тех случаев, когда эти требования также включены в 6.1, 7.2, 8.3, 8.4 или 8.5. Заявления, содержащие слова «должен», «должна», «должны» и пр., являются рекомендательными, но не обязательными.

2.2 Соответствие продуктов

2.2.1 Примеры причин необходимости обеспечения соответствия продуктов

В организации могут иметься несколько причин, по которым организация может пожелать обеспечить соответствие отдельных продуктов данной части настоящего стандарта. Например, обеспечение соответствия может потребоваться, когда конкретный продукт поставляется на рынок, требующий соблюдения соответствия (например, если государственные организации требуют, чтобы продукт соответствовал данной части настоящего стандарта с целью включения в проект). Обеспечения соответствия также могут требовать провайдеры платформы, желающие организовать более безопасное и контролируемое хранилище тегов, которое может использоваться для однозначной идентификации того, какой именно конечный пользователь установил какой именно программный пакет.

2.2.2 Область действия продукта

Должно быть оформлено четко выраженное заявление об области действия продукта, однозначно описывающее программные продукты, к которым применяется данная область действия и, в соответствующих случаях, вносящее уточнения относительно продуктов, к которым данная область действия не применяется. Область действия соответствия продукта может определяться любым удобным способом, например, для конкретного программного продукта, для всех программных продуктов, для всех программных продуктов, установленных на конкретных платформах, для программных продуктов конкретных изготовителей и/или для всех программных продуктов, созданных после указанной даты, при условии, что такие определения области действия являются однозначными. В случае продукта, создающего или изменяющего теги идентификации программного обеспечения, областью действия будет считаться сам продукт и все программное обеспечение, произведенное или измененное продуктом при включенной функциональности обеспечения соответствия тегов.

2.2.3 Соответствие программных продуктов

Полное соответствие программного продукта достигается одним из двух способов:

a) Для продукта, предназначенного для установки, полное соответствие достигается посредством демонстрации того, что все теги идентификации программного обеспечения, установленные этим продуктом в процессе установки, соответствуют всем обязательным требованиям данной части настоящего стандарта, определенным в 6.1 и 8.3. Если используются дополнительные или расширенные элементы тегов, такие элементы тегов также должны соответствовать требованиям, определенным в 8.4 и 8.5.

Такое соответствие должно быть продемонстрировано посредством выполнения эквивалентного разбиения с критерием завершения, состоящим в том, что пройдены все тесты и достигнуто 100 % эквивалентное разбиение в процессе создания/установки тега. Эквивалентные разбиения должны определяться на основании заявления об области действия продукта. Если программный продукт состоит из пакета других программных продуктов, то в программном продукте должны быть сохранены все теги компонентов и содержаться ссылки на все элементы дочерних тегов, которые при любых обстоятельствах по-прежнему должны идентифицироваться отдельно (с целью лицензирования, обеспечения безопасности или с другими целями);

b) для продукта, предназначенного для распространения, но еще не установленного, полное соответствие достигается посредством демонстрации того, что распространяемые сборки выпущены с уникальным тегом, который должен соответствовать всем обязательным требованиям данной части настоящего стандарта, определенным в 6.1 и 8.3. Если используются дополнительные или расширенные элементы тегов, такие элементы тегов также должны соответствовать требованиям, определенным в 8.4 и 8.5. Исключение: любые обязательные элементы, присущие конкретной системе, не включаются.

Такое соответствие должно быть продемонстрировано посредством выполнения эквивалентного разбиения с критерием завершения, состоящим в том, что пройдены все тесты и достигнуто 100 % эквивалентное разбиение. Эквивалентные разбиения должны определяться на основании заявления об области действия продукта.

Если программный продукт состоит из пакета других программных продуктов, то в программном продукте должны сохраниться все теги компонентов и содержаться ссылки на все элементы дочерних тегов, которые при любых обстоятельствах по-прежнему должны идентифицироваться отдельно (с целью лицензирования, обеспечения безопасности или с другими целями).

2.2.4 Соответствие тегов идентификации программного обеспечения третьих сторон

Сторонние организации, предоставляющие теги, могут применять процессы создания тегов идентификации программного обеспечения для любых программных пакетов, не содержащих такие теги. Такие процессы могут применяться для программных продуктов более ранних версий, продуктов типа shareware/freeware или для для продуктов компаний, принявших решение не обеспечивать соответствие данной части настоящего стандарта. Такие теги могут предоставляться организациям для того, чтобы они могли обнаруживать программное обеспечение и запускать процедуры идентификации.

Полное соответствие тегов идентификации программного обеспечения, созданных третьими сторонами, достигается посредством демонстрации того, что все теги идентификации программного обеспечения, созданные организацией, соответствуют всем обязательным требованиям данной части настоящего стандарта, определенным в 6.1 и 8.3. Если используются дополнительные или расширенные элементы тегов, такие элементы тегов также должны соответствовать требованиям, определенным в 8.4 и 8.5. Любые добавляемые новые данные должны соответствовать тем же стандартам, которые применяются к обеспечению соответствия устанавливаемого программного обеспечения.

Обеспечение соответствия тегов идентификации программного обеспечения, созданных третьими сторонами, требует, чтобы поставщики тегов продемонстрировали уникальность созданных ими идентификаторов программного обеспечения (software_id), и что для идентификации поставщиков программного обеспечения в таких идентификаторах используются согласованные значения. Предполагается, что поставщики тегов будут вести список уникальных поставщиков программного обеспечения для всех созданных тегов, причем такой список будет включать в себя согласованный регистрационный идентификатор (regid) поставщика программного обеспечения (являющийся ссылкой на домен поставщика) и уникальный идентификатор ID (которым может быть идентификатор GUID) для каждой ссылки, и эти сведения на согласованной основе будут использоваться в созданных тегах.

Такое соответствие должно быть продемонстрировано посредством выполнения эквивалентного разбиения с критерием завершения, состоящим в том, что пройдены все тесты и достигнуто 100 % эквивалентное разбиение в процессе создания тега. Эквивалентные разбиения должны определяться как по диапазону программного обеспечения, к которому будет применяться инструментарий для работы с тегами, так и по соответствующим заявлениям об области действия продукта.

2.2.5 Соответствие продуктов установки программного обеспечения

Полное соответствие продукта установки программного обеспечения достигается посредством демонстрации того, что все установленные им в процессе установки теги идентификации программного обеспечения соответствуют всем обязательным требованиям данной части настоящего стандарта, определенным в 6.1 и 8.3. Если используются дополнительные или расширенные элементы тегов, такие элементы тегов также должны соответствовать требованиям, определенным в 8.4 и 8.5.

Такое соответствие должно быть продемонстрировано посредством выполнения эквивалентного разбиения с критерием завершения, состоящим в том, что пройдены все тесты и достигнуто 100 % эквивалентное разбиение в процессе создания/установки тега. Эквивалентные разбиения должны определяться как по диапазону устанавливаемого программного обеспечения, так и по соответствующим заявлениям об области действия продукта.

Если устанавливаемый программный продукт состоит из пакета других программных продуктов, то в программном продукте должны быть сохранены все теги компонентов и содержаться ссылки на все элементы дочерних тегов, которые при любых обстоятельствах по-прежнему должны идентифицироваться отдельно (с целью лицензирования, обеспечения безопасности или с другими целями).

Значения существующих тегов, поставляемых вместе с распространяемым программным обеспечением, не должны меняться ни в каком случае (за некоторыми особыми исключениями). Если распространенный тег идентификации программного обеспечения окажется поврежденным, и в этом теге идентификации программного обеспечения не предусмотрен порядок «валидации» для внесения исправлений в тег, в программном продукте могут быть предусмотрены способы обработки таких типов исключительного состояния, разрешаемые к применению специалистом-практиком по использованию процессов SAM. На основании действий, определяемых специалистом-практиком по использованию процессов SAM, обработка таких исключительных состояний может включать в себя такие действия, как внесение исправлений в тег идентификации программного обеспечения в случае его повреждения, удаление тега идентификации программного обеспечения, если он более не принадлежит устройству, или внесение изменений в тег идентификации программного обеспечения, чтобы указать, что программное обеспечение больше не установлено на устройстве. При внесении пользователем любых изменений в тег такие действия пользователя должны фиксироваться, и сведения о них должны сохраняться в программном продукте.

Предполагается, что такие продукты будут иметь возможность включения или отключения данной функциональности. Заявление о соответствии продукта применяется только к продукту, на котором данная функциональность включена.

2.2.6 Соответствие инструментария для работы с тегами

Полное соответствие инструментария для работы с тегами достигается одним из двух способов:

а) полное соответствие инструментария для работы с тегами, устанавливающего или вносящего изменения в установленные теги идентификации программного обеспечения, достигается посредством демонстрации того, что все теги идентификации программного обеспечения, установленные или измененные продуктом, соответствуют всем обязательным требованиям данной части настоящего стандарта, определенным в 6.1 и 8.3. Если используются дополнительные или расширенные элементы тегов, такие элементы тегов также должны соответствовать требованиям, определенным в 8.4 и 8.5. Любые добавляемые новые данные должны соответствовать тем же стандартам, которые применяются к обеспечению соответствия устанавливаемого программного обеспечения.

Такое соответствие должно быть продемонстрировано посредством выполнения эквивалентного разбиения с критерием завершения, состоящим в том, что пройдены все тесты и достигнуто 100 % эквивалентное разбиение в процессе создания тега. Эквивалентные разбиения должны определяться как по диапазону программного обеспечения, к которому будет применяться инструментарий для работы с тегами, так и по соответствующим заявлениям об области действия продукта.

Если устанавливаемый программный продукт состоит из пакета других программных продуктов, то в программном продукте должны быть сохранены все теги компонентов и содержаться ссылки на все

элементы дочерних тегов, которые при любых обстоятельствах по-прежнему должны идентифицироваться отдельно (с целью лицензирования, обеспечения безопасности или с другими целями).

Значения существующих тегов, поставляемых вместе распространяемым программным обеспечением, не должны меняться ни в каком случае (за некоторыми особыми исключениями). Если распространенный тег идентификации программного обеспечения окажется поврежденным, и в этом теге идентификации программного обеспечения не предусмотрен порядок «валидации» для внесения исправлений в тег, в программном продукте могут быть предусмотрены способы обработки таких типов исключительного состояния, разрешаемые к применению специалистом-практиком по использованию процессов SAM. На основании действий, определяемых специалистом-практиком по использованию процессов SAM, обработка таких исключительных состояний может включать в себя такие действия, как внесение исправлений в тег идентификации программного обеспечения в случае его повреждения, удаление тега идентификации программного обеспечения, если он более не принадлежит устройству, или внесение изменений в тег идентификации программного обеспечения, чтобы указать, что программное обеспечение больше не установлено на устройстве. При внесении пользователем любых изменений в тег такие действия пользователя должны фиксироваться, и сведения о них должны сохраняться в программном продукте.

Предполагается, что такие продукты будут иметь возможность включения или отключения данной функциональности. Заявление о соответствии продукта применяется только к продукту, на котором данная функциональность включена;

b) для инструментария для работы с тегами, который обнаруживает теги, собирает теги, составляет отчеты по тегам и использует теги (например, инструментария для обнаружения тегов, инструментов управления настольными системами или инструментов выверки SAM), полное соответствие достигается посредством демонстрации следующего.

1) демонстрации того, что собраны все теги, доступные на вычислительном устройстве. К таким тегам относятся теги, хранящиеся в общих системных каталогах, а также теги, расположенные в каталогах верхнего уровня установленного программного обеспечения,

2) демонстрации того, что все теги, собранные с вычислительных устройств и хранящиеся в хранилище инструментария по работе с тегами, содержат точно такую же информацию, что и содержимое тега, расположенного на вычислительном устройстве, с которого он был первоначально получен,

3) если тег имеет цифровую подпись, и имеется соответствующий публичный ключ, демонстрации того, что инструментарий подтверждает подпись, и что информация была подписана.

Такое соответствие должно быть продемонстрировано посредством выполнения эквивалентного разбиения с критерием завершения, состоящим в том, что пройдены все тесты и достигнуто 100 % эквивалентное разбиение в процессе сбора/валидации тегов. Эквивалентные разбиения должны определяться как по диапазону программного обеспечения, анализ которого должен провести инструментарий инструментарий для работы с тегами, так и по соответствующим заявлениям об области действия продукта.

Если устанавливаемый программный продукт состоит из пакета других программных продуктов, то в программном продукте должны быть сохранены все теги компонентов и содержаться ссылки на все элементы дочерних тегов, которые при любых обстоятельствах по-прежнему должны идентифицироваться отдельно (с целью лицензирования, обеспечения безопасности или с другими целями).

Значения существующих тегов, поставляемых вместе распространяемым программным обеспечением, не должны меняться ни в каком случае (за некоторыми особыми исключениями). Если распространенный тег идентификации программного обеспечения окажется поврежденным, и в этом теге идентификации программного обеспечения не предусмотрен порядок «валидации» для внесения исправлений в тег, в программном продукте могут быть предусмотрены способы обработки таких типов исключительного состояния, разрешаемые к применению специалистом-практиком по использованию процессов SAM. На основании действий, определяемых специалистом-практиком по использованию процессов SAM, обработка таких исключительных состояний может включать в себя такие действия, как внесение исправлений в тег идентификации программного обеспечения в случае его повреждения, удаление тега идентификации программного обеспечения, если он более не принадлежит устройству, или внесение изменений в тег идентификации программного обеспечения, чтобы указать, что программное обеспечение больше не установлено на устройстве. Если конечный пользователь вносит любые изменения в тег, такие действия конечного пользователя должны фиксироваться, и сведения о них должны сохраняться в программном продукте.

Предполагается, что такие продукты будут иметь возможность включения или отключения данной функциональности. Заявление о соответствии продукта применяется только к продукту, на котором данная функциональность включена.

2.2.7 Соответствие платформы

Полное соответствие функциональности тегов платформы достигается посредством демонстрации того, что платформа способна централизованно хранить данные тегов идентификации программного обеспечения и обеспечивать действенное функционирование сервисов, оговоренных в 7.2:

a) базовая функциональность: добавление, изменение, считывание и удаление данных тега;

b) безопасность: определение того, каким именно конечным пользователям разрешается считывать, создавать, удалять и вносить изменения в теги идентификации программного обеспечения;

c) функциональность аудита: идентификация того, какой именно конечный пользователь установил, внес изменения или удалил данный элемент конфигурации программного обеспечения, и когда произошли эти изменения.

Такое соответствие должно быть продемонстрировано посредством выполнения эквивалентного разбиения с критерием завершения, состоящим в том, что пройдены все тесты и достигнуто 100 % эквивалентное разбиение хранилища тегов. Эквивалентные разбиения должны определяться как по диапазону программного обеспечения, которое должно размещаться на платформе, так и по соответствующим заявлениям об области действия продукта.

2.3 Организационное соответствие

2.3.1 Примеры причин необходимости обеспечения организационного соответствия

Организации могут пожелать обеспечить соответствие данной части настоящего стандарта по нескольким причинам. Например, поставщики программного обеспечения могут пожелать популяризировать свои программные продукты как более простые и удобные в управлении. Кроме того, потребители программного обеспечения могут пожелать продемонстрировать, что они осуществляют активное управление своими программными активами, и что они могут предоставить точную информацию по любым запросам на проведение сверок или проверок.

2.3.2 Организационная область действия

Должно быть оформлено четко выраженное заявление об организационной области действия, однозначно описывающее организационную структуру, к которой применяется данная область действия и, в соответствующих случаях, вносящее уточнения относительно сегментов, к которым данная область действия не применяется. Заявление об организационной области действия должно сопровождаться заявлением об области действия продукта.

2.3.3 Соответствие поставщика программного обеспечения

Полное соответствие поставщика программного обеспечения достигается посредством демонстрации организацией того, что все программное обеспечение, относящееся к области действия, отвечает соответствующим требованиям к соответствию продукта, оговоренным в 2.2.3.

2.3.4 Соответствие поставщика инструментария для работы с тегами

Полное соответствие поставщика инструментария для работы с тегами достигается посредством демонстрации организацией того, что все программное обеспечение, относящееся к области действия, отвечает соответствующим требованиям к соответствию инструментария, оговоренным в 2.2.6.

Кроме того, для того чтобы иметь возможность подачи заявления о соответствии поставщика инструментария для работы с тегами, весь инструментарий для работы с тегами, изготовленный организацией, должен быть включен в область действия продукта.

2.3.5 Соответствие потребителя программного обеспечения

Полное соответствие организации, устанавливающей программное обеспечение, достигается посредством демонстрации того, для всего программного обеспечения, относящегося к области действия продукта организации потребителя программного обеспечения, имеются все теги идентификации программного обеспечения, и что теги идентификации программного обеспечения соответствуют всем

обязательным требованиям данной части настоящего стандарта, определенным в 6.1 и 8.3. Если используются дополнительные или расширенные элементы тегов, такие элементы тегов также должны соответствовать требованиям, определенным в 8.4 и 8.5.

2.4 Соответствие соглашения

Данная часть настоящего стандарта может использоваться для разработки соглашения между поставщиком программного обеспечения и потребителем программного обеспечения, и в этом случае в такое соглашение могут быть перенесены некоторые пункты данной части настоящего стандарта (с изменениями или без изменений). В этом случае требуется, чтобы обе стороны соблюдали условия соглашения между ними, а не условия данной части настоящего стандарта.

Примечание – На данную часть настоящего стандарта и его содержание распространяются авторские права и патентная политика ИСО/МЭК. Тем не менее, в случае обеспечения соответствия соглашения получение разрешения на использование авторского права не является необходимым.

3 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие документы. Для датированных ссылок применяется только указанная версия документа. Для недатированных ссылок применяется последняя версия ссылочного документа (со всеми изменениями и дополнениями).

Унифицированный идентификатор ресурса: Общий синтаксис, 2005 (IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax)

Теги для идентификации языков, 2006 (IETF RFC 4646, Tags for Identifying Languages)

Правила обработки и синтаксиса цифровой подписи XML (второе издание), 2008 (W3C Recommendation, XML Signature Syntax and Processing) (Second Edition)

Схема XML Часть 2: Типы данных, 2004 (W3C Recommendation, XML Schema Part 2: Datatypes) (Second Edition)

Система сиандартных продуктов и услуг ООН (UNSPSC, The United Nations Standard Products and Services Code)

4 Термины, определения и сокращения

4.1 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

4.1.1 **приложение** (application): Система, предназначенная для сбора, сохранения, обработки и представления данных посредством компьютера.

Примечание – Термин «приложение» в общем случае применяется к компонентам программного обеспечения, которые могут запускаться на выполнение.

4.1.2 **комплект** (bundle): Группа продуктов, формируемых в результате применения стратегии маркетинга/лицензирования с целью продажи прав на использование нескольких продуктов как одного приобретаемого предмета.

Примечания

1 Комплект может называться «набором», если продукты тесно связаны друг с другом и, в общем случае, интегрированы друг в друга (например, комплект офисных продуктов может содержать программное обеспечение для работы с электронными таблицами, текстовыми документами, презентациями и другими аналогичными структурами).

2 Комплектами также могут называться наименования программных продуктов, связанных между сбой менее тесным образом (например, компьютерная игра, сканер вирусов и антивирусная утилита могут быть оформлены в «комплект» для поставки вместе с новым компьютером), или группы прав на использование (например, набор прав на использование продуктов резервного копирования программного обеспечения).

4.1.3 **компонент** (component): Объект дискретной структуры, например сборка или программный модуль, в составе системы, рассматриваемый на конкретном уровне анализа.

Примечание – Под компонентом понимается часть целого, например, компонент программного продукта, компонент тега идентификации программного обеспечения и пр.

4.1.4 **вычислительное устройство** (computing device): Функциональный блок, способный выполнять достаточно сложные вычислительные операции, в том числе различные арифметические и логические операции, без человеческого вмешательства.

Примечание – Вычислительное устройство может состоять из автономно функционирующего блока или нескольких взаимосопряженных блоков. Вычислительным устройством также может являться устройство, реализующее специальный набор функций, например, телефон или карманный персональный компьютер, или реализующее функции более общего характера, например, переносной или настольный компьютер.

4.1.5 **элемент конфигурации** (configuration item, CI): Отдельный элемент или объединенные части оборудования или программного обеспечения, или совокупность и того, и другого, управление которыми осуществляется как отдельным объектом.

Примечание – Элементы конфигурации могут отличаться друг от друга по сложности, размеру и типу. Элементом конфигурации, в частности, может быть вся система, в том числе все оборудование, программное обеспечение и документация, а также любой отдельный модуль или мелкий аппаратный компонент или отдельный программный пакет.

4.1.6 **база данных управления конфигурацией** (configuration management database, CMDB): База данных, в которой содержатся все соответствующие сведения о каждом элементе конфигурации и сведения о важных взаимосвязях между ними.

4.1.7 **клиент** (customer): Конечные пользователи или организации, для которых издатель программных продуктов проектирует и разрабатывает программное обеспечение и продает права на использование такого программного обеспечения.

4.1.8 **элемент** (element): Компонент тега идентификации программного обеспечения, выдающий информацию, относящуюся к программному обеспечению, представленному тегом.

Примечание – Различные типы элементов определены в 8.3, 8.4 и 8.5.

4.1.9 **конечный пользователь** (end-user): Лицо (или лица), которое для управления программными пакетами или использования программных пакетов работает непосредственно на вычислительном устройстве или взаимодействует непосредственно с вычислительным устройством.

4.1.10 **эквивалентное разбиение** (equivalence partitioning): Процесс тестирования программного обеспечения, идентифицирующий репрезентативные группы входных значений, обрабатываемых программными продуктами как обычно, позволяющий осуществлять выборку из каждой репрезентативной группы с целью проверки правильности выходных данных, что позволяет сократить количество сценариев тестирования с сохранением полного покрытия всех сценариев тестирования.

4.1.11 **расширяемый язык разметки** (extensible markup language, (XML): Не привязанный к определенной платформе язык разметки, для использования которого не требуются лицензии, реализующий правила генерации текстовых форматов, содержащих структурированные данные.

4.1.12 **глобальный уникальный идентификатор** (globally unique identifier, GUID): 16-байтовая строка символов, формируемая способом, обеспечивающим высокую вероятность того, что строка является уникальной в любом контексте.

Примечания

1 В ряде случаев могут использоваться другие алгоритмы формирования глобальных уникальных идентификаторов. В общем случае альтернативные алгоритмы используют структуры на базе универсального кода ресурса (Uniform Resource Identifier, URI), и, соответственно, в глобальный уникальный идентификатор должен быть включен регистрационный идентификатор владельца id (regid).

2 Обозначение GUID (в виде сокращения из заглавных букв) означает 16-байтовую версию. Если термин указан прописными буквами (guid), это означает, что общий алгоритм может использовать либо идентификатор URI, либо 16-байтовый идентификатор.

4.1.13 **устаревшее программное обеспечение** (legacy software): Программное обеспечение, исходно созданное без тегов идентификации программного обеспечения.

4.1.14 **разработчик приложений для направления бизнеса** (line of business application developer): Лицо или компания, специализирующиеся на разработке приложений, реализующих конкретные функции для выполнения конкретных бизнес-задач.

4.1.15 **алгоритм хэширования Message-Digest 5** (MD5, Message-Digest algorithm 5): Алгоритм, реализующий широко распространенную криптографическую хэш-функцию с 128-разрядным хэш-значением. Этот алгоритм часто используется с целью идентификации файлов, содержащих одинаковые данные.

4.1.16 **пакет** (package): Набор связанных компонентов, объединенных в единый распространяемый элемент.

Примечание – Например, программным пакетом может быть набор файлов, которые могут использоваться для установки программного обеспечения на вычислительное устройство и которые могут распространяться через CD или электронными средствами.

4.1.17 **платформа** (platform). Компьютер, или аппаратное устройство, и/или связанная операционная система, или виртуальная среда, на которых может устанавливаться или запускаться программное обеспечение.

Примечание – Примеры платформ: Linux™, Microsoft Vista® и Java™.

4.1.18 **провайдер платформы** (platform provider): Организация, отвечающая за функционирование платформы.

Примечание – В общем случае в качестве провайдера платформы выступает поставщик соответствующей операционной системы или виртуальной среды.

4.1.19 **продукт** (product): Полный комплект компьютерных программ, процедур, связанной документации и данных, предназначенных для поставки программного обеспечения потребителю.

Примечание – Термины «продукт» и «программный пакет» могут использоваться на равных основаниях, в зависимости от контекста описываемого элемента.

4.1.20 **регистрационный идентификатор** (registration identifier, (regid): Идентификатор, создаваемый по доменному имени и дате, в которую домен был приобретен конкретным физическим лицом или компанией. Такой идентификатор позволяет физическому лицу или компании иметь собственное уникальное пространство имен и самим выступать в качестве органа, регистрирующего все публикуемые ими элементы конфигурации программного обеспечения без обращения к специальным отраслевым регистрационным органам.

4.1.21 **релиз** (release): Набор новых и/или измененных элементов конфигурации, которые были совместно протестированы и внедрены в производственное окружение.

4.1.22 **администратор релизов** (release manager): Физическое лицо, отвечающее за управление набором новых и/или измененных элементов конфигурации, которые были совместно протестированы и внедрены в реальное производственное окружение организации.

4.1.23 **владелец процессов SAM** (SAM owner): Физическое лицо на высшем уровне руководства организации, определенное как лицо, ответственное за процессы SAM.

4.1.24 **специалист-практик по использованию процессов SAM** (SAM practitioner): Физическое лицо, занимающееся на практике или в должностные обязанности которого входит управление программными активами.

Примечание – Специалист-практик по использованию процессов SAM часто привлекается к выполнению задач инвентаризации программного обеспечения и/или определения прав на использование программного обеспечения.

4.1.25 **программное обеспечение** (software): Все или часть программ, процедур, правил и связанной документации, относящихся к системе обработки информации.

4.1.26 **управление программными активами** (Software Asset Management, SAM): Эффективное использование, контроль и защита программных активов в рамках организации.

4.1.27 **потребитель программного обеспечения** (software consumer): Организация или лицо, приобретающее права на использование программного пакета.

4.1.28 **создатель программного обеспечения** (software creator): Лицо или организация, создающее программный продукт или программный пакет.

Примечание – Такое лицо может использовать или не использовать права на продажу или распространение программного обеспечения.

4.1.29 **разработчик программного обеспечения** (software developer): Лицо, создающее программное обеспечение, которое может выполнять определенный набор действий.

Примечание – Часто разработчик программного обеспечения работает вместе с другими разработчиками для производителя программного обеспечения с целью создания коммерческих приложений. Разработчик программного обеспечения может также выступать в качестве разработчика программного обеспечения для внутренних нужд организации, в которой работает разработчик программного обеспечения.

4.1.30 **право на использование программного обеспечения** (software entitlement): Юридическое право собственности в отношении прав на использование программных лицензий, определяемое посредством заключения соглашений между покупателем программного обеспечения и владельцем авторских прав на программное обеспечение.

Примечание – Права на эффективное использование учитывают все контракты и действующие лицензии, в том числе полные лицензии, обновления лицензий и соглашения и техническом обслуживании.

4.1.31 **тег идентификации программного обеспечения** (software identification tag): Файл, составленный из обязательных элементов, необязательных элементов и дополнительных данных, содержащий достоверную идентификационную информацию об элементе конфигурации программного обеспечения.

Примечание – Описание обязательных элементов приведено в 8.3, описание необязательных элементов приведено в 8.4, описание дополнительной информации приведено в 8.5.

4.1.32 **программная лицензия** (software license): Юридические права на использование программного обеспечения в соответствии с условиями, оговоренными владельцем авторских прав на программное обеспечение.

Примечание – Под термином «использование программного продукта» может пониматься: доступ, копирование, распространение, установка и запуск на выполнение программного продукта, в зависимости от условий использования, оговоренных для этого продукта.

4.1.33 **лицензиар программного обеспечения** (software licensor): Лицо или организация, владеющие правами на выпуск программной лицензии для конкретного программного пакета.

4.1.34 **производитель программного обеспечения** (software manufacturer): Группа людей или организация, разрабатывающая программное обеспечение в общем случае для его распространения и использования другими людьми или организациями.

4.1.35 **программный пакет** (software package): Законченный и документированный набор программ, поставляемый для конкретного применения или функции.

Примечание – В данной части настоящего стандарта термин «программный пакет» означает набор файлов, связанных с конкретным набором бизнес-функций, которые могут быть установлены на вычислительном устройстве, и к которым применяется набор конкретных лицензионных требований. В данной части настоящего стандарта термины «продукт» и «программный пакет» используются как синонимы и применяются в зависимости от контекста описываемого элемента.

4.1.36 **упаковщик программного обеспечения** (software packager): Лицо, осуществляющий перекомпоновку или объединение в пакеты программного обеспечения, созданного другими лицами.

Примечание – Такая упаковка может осуществляться продавцом, создающим добавочную стоимость товара, объединяющим программные пакеты в единый пакет, который будет работать в составе встроенной системы, или реселлером программного обеспечения, имеющим лицензию на объединение некоторого количества различных программных продуктов в один пакет.

4.1.37 **поставщик программного обеспечения** (software provider): Лицо, которое создает (создатель программного обеспечения), изменяет (модификатор программного обеспечения) или лицензирует (лицензиар программного обеспечения) программное обеспечение для распространения или установки.

Примечание – К таким лицам относятся производители программного обеспечения, независимые разработчики программного обеспечения, консультанты и переупаковщики ранее произведенного программного обеспечения. ИТ-департаменты также могут считаться разработчиками программного обеспечения для внутренних целей организации.

4.1.38 **издатель программного обеспечения** (software publisher): Лицо, группа или компания, осуществляющие упаковку и распространение программного обеспечения. Издатели программного обеспечения могут являться или не являться производителем программного обеспечения.

4.1.39 **создатель тега** (tag creator): Лицо, первоначально создающее тег идентификации программного обеспечения.

Примечание – Данное лицо может являться частью организации, создавшей программное обеспечение, и в этом случае создателем тега и создателем программного обеспечения являются одни и те же лица. Создателем тега также может быть сторонняя организация, не имеющая отношения к создателю программного обеспечения (например, если теги создаются для устаревшего программного обеспечения).

4.1.40 **модификатор тега** (tag modifier): Упаковщик программного обеспечения или потребитель программного обеспечения, вносящий изменения в тег после его создания.

Примечание – Внесение изменений в любой тег ограничивается элементами, изменение которых было разрешено лицензиаром программного обеспечения, и осуществляется на базе лицензионных или договорных соглашений с создателем тега и/или создателем программного обеспечения. Модификатору тега разрешается добавлять значения в тег идентификации программного обеспечения (например, реселлер может добавлять сведения о том, где именно был приобретен продукт), или вносить изменения в существующие части тега (например, продавец, создающий добавочную стоимость товара, может создавать набор программных продуктов, который будет выглядеть как изготовленный в одном месте и одним лицом).

4.1.41 **поставщик тегов** (tag provider): Лицо, которое создает (создатель тега), изменяет (модификатор тега) теги идентификации программного обеспечения для программных пакетов.

Примечание – Поставщик тегов может являться частью организации поставщика программного обеспечения или может быть сторонней организацией или потребителем программного обеспечения.

4.1.42 **унифицированный идентификатор ресурса** (Uniform Resource Identifier, URI): Компактная последовательность символов, идентифицирующих абстрактный объект или физический ресурс, доступный в сети Интернет.

Примечание – Синтаксис, используемый для формирования URI, определен в стандарте IETF RFC 3986.

4.1.43 **действующий [тег]** (valid): Данные тега идентификации программного обеспечения <XML-файл> соответствуют определению XSD, и тег идентификации программного обеспечения является действующим с точки зрения XML.

Примечание – См. также 4.1.44.

4.1.44 **действующий** (valid): Процесс <тег идентификации программного обеспечения>, используемый для обеспечения правильности данных, включенных в установленный тег идентификации программного обеспечения.

Примечание – См. также 4.1.43.

4.1.45 **продавец, создающий добавочную стоимость товара** (value-added reseller): Компания, имеющая лицензию на переупаковку и поддержку существующих продуктов в виде объединенных программных пакетов.

4.1.46 **версия** (version): Уникальная буквенно-цифровая строка, обозначающая уникальную версию элемента.

Примечание – Версии часто используются в программном обеспечении для идентификации переработанного программного обеспечения, обеспечивающего уникальную функциональность или исправление ошибок. Версии в общем случае могут состоять из нескольких частей, причем основной номер версии означает существенные изменения в функциональности или пользовательского интерфейса, а дополнительный номер версии означает менее существенные изменения функциональности или пользовательского интерфейса.

4.1.47 **Определение схемы XML** (XML Schema Definition): Яык на базе XML, оговаривающий набор правил и структуру создания документов XML.

Примечание – Для того, чтобы считаться «действительными» документами, документы XML должны быть составлены в соответствии с правилами, установленными в определении XSD.

4.2 Сокращения

CI	configuration item	Элемент конфигурации
CMDB	configuration management database	База данных управления конфигурацией
GUID	globally unique identifier	Глобальный уникальный идентификатор
IETF	Internet Engineering Task Force	Рабочая группа по стандартам для сети Интернет
MD5	message digest 5	Алгоритм message digest 5
regexp	regular expression	Регулярное выражение
regid	registration identifier	Регистрационный идентификатор
SAM	software asset management	Управление программными активами
URI	uniform resource identifier	Унифицированный идентификатор ресурса

URL	uniform resource locator	Унифицированный указатель ресурса
VAR	value added reseller	Продавец, создающий добавочную стоимость товара
W3C	World Wide Web Consortium	Консорциум W3C
XML	Extensible Markup Language	Расширяемый язык разметки
XSD	XML Schema Definition	Определение схемы XML

5 Согласование и координация с предыдущими стандартами

5.1 Заявление о согласовании с данной частью настоящего стандарта

Содержание данной части настоящего стандарта предназначено для дополнения предыдущих публикаций ИСО/МЭК 19770 и должно быть согласовано с ними.

5.2 Согласование с ИСО/МЭК 19770-1:2006 Информационная технология. Управление программными активамию Часть 1. Процессы

Данная часть настоящего стандарта согласована со следующими частями ИСО/МЭК 19770-1:2006:

a) ИСО/МЭК 19770-1:2006, 3, в котором оговариваются условия и определения, относящиеся к этому документу.

Данная часть настоящего стандарта согласована с ИСО/МЭК 19770-1:2006, условия и определения, содержащиеся в 3, относящиеся к обеим частям, воспроизведены в настоящем документе;

b) в ИСО/МЭК 19770-1:2006, 4.4.2.2, оговорено следующее: «Реализация процесса «Идентификация программных активов» позволит организации продемонстрировать, что: a) в организации официально определены типы контролируемых активов и связанная с ними информация, b) в организации имеется реестр хранилищ и проинвентаризированных объектов, содержащий сведения о хранимых инвентарных запасах и типах информации. Дублирование информации такого реестра допускается только в том случае, если дублируемую информацию можно отследить до определенной исходной записи».

Данная часть настоящего стандарта подтверждает необходимость официальных определений, наличия хранилищ и проинвентаризированных объектов для элементов конфигурации программного обеспечения. Согласно ИСО/МЭК 19770-1:2006, 4.4.2.2.a.2.i, к элементам конфигурации программного обеспечения относятся «все платформы, на которых может устанавливаться или запускаться программное обеспечение». В настоящем стандарте отсутствует явное требование для элемента «платформа», так как в данной части настоящего стандарта внимание сосредоточено на идентификации программного обеспечения, обнаруженного на вычислительных устройствах, а не на требованиях к общему хранилищу программного обеспечения. Предполагается, что инструментарий для обнаружения тегов будет собирать информацию о платформе в ходе процесса инвентаризации;

c) в ИСО/МЭК 19770-1:2006, 4.4.2.2, оговорено следующее: «Базовая информация для всех активов: i) Уникальный идентификатор, ii) Имя/описание, iii) Расположение, iv) Ответственный (или владелец), v) Статус (например, тестовый/производственный; разработка или сборка), vi) Тип (например, программное обеспечение, аппаратное обеспечение, устройство), vii) Версия (если имеется)».

Данная часть настоящего стандарта подтверждает эти требования в отношении базовой информации. Тем не менее, в данную часть настоящего стандарта базовая информация «Расположение» и «Ответственный» не включается в качестве значений элемента конфигурации программного обеспечения, поскольку эта информация связана с активом, на котором происходит обнаружение элемента конфигурации программного обеспечения, а не с самим элементом.

Статус элемента конфигурации программного обеспечения определяется значениями Release (Релиз) в теге идентификации программного обеспечения. Эти значения не являются обязательными. Рекомендуется предоставление этих значений вместе с информацией, относящейся к дате утверждения и оператору, выполнившему процесс (см. 8.4).

5.3 Согласование с ИСО/МЭК 20000-1:2005 Информационная технология. Управление услугами. Часть 1. Спецификации

a) В ИСО/МЭК 20000-1:2005, пункт 9.1, оговаривается следующее: «При необходимости должна быть предусмотрена возможность отслеживания и проверки элементов конфигурации на изменения, например, изменений и перемещений программного и аппаратного обеспечения». Данная часть

настоящего стандарта подтверждает использование тегов идентификации программного обеспечения для обнаружения и определения доступной для отслеживания и проверки информации по элементам конфигурации программного обеспечения.

b) В ИСО/МЭК 20000-1:2005, 9.1, оговаривается следующее: «Процедуры управления конфигурацией должны обеспечивать сохранение целостности систем, сервисов и сервисных компонентов».

Данная часть настоящего стандарта утверждает необходимость в процедурах управления конфигурацией для обеспечения целостности, и, таким образом, предоставляет создателям возможность включения в продукт цифровой подписи (см. 6.1.11). Цифровая подпись может использоваться для проверки того, что определенные значения обязательных элементов не подверглись изменениям. Такая проверка, соответственно, позволит поставщикам программного обеспечения или поставщикам тегов достоверно выявлять случаи взлома тегов идентификации программного обеспечения или отсутствие тегов.

c) В ИСО/МЭК 20000-1:2005, 9.1, оговаривается следующее: «Управление мастер-копиями цифровых элементов конфигурации должно осуществляться в защищенных физических или электронных библиотеках, и ссылки на них должны присутствовать в записях конфигурации, например, программного обеспечения, результатов тестирования, вспомогательной документации».

Данная часть настоящего стандарта рекомендует включение тегов идентификации программного обеспечения в соответствующие элементы конфигурации программного обеспечения в библиотеку эталонного программного обеспечения.

d) В ИСО/МЭК 20000-1:2005, 9.1, оговаривается следующее: «Все элементы конфигурации должны уникально идентифицироваться и фиксироваться в базе данных CMDB, доступ к которой должен быть строго контролируем».

Данная часть настоящего стандарта согласуется с тем требованием, что все элементы конфигурации программного обеспечения должны уникально идентифицироваться. Элемент конфигурации программного обеспечения уникально идентифицируется идентификатором продукта, серийным номером, единицей складского хранения, каждый из которых можно соотнести с подтверждением наличия лицензии, заказом на покупку и элементами конфигурации программного обеспечения, хранящимися в базе данных CMDB (см. 8.4.14, 8.4.20, 8.4.21).

Метод, в соответствии с которым тег идентификации программного обеспечения сохраняется в базе данных CMDB и получает уникальный ссылочный номер в этой базе данных, в данной части настоящего стандарта (см. 1.2) не рассматривается.

d) В ИСО/МЭК 20000-1:2005, 10.1, оговаривается следующее: «Процессы выпуска релизов и распространения должны разрабатываться и реализовываться таким образом, чтобы в процессе установки, обработки, упаковки и поставки сохранялась целостность аппаратного и программного обеспечения».

Данная часть настоящего стандарта подтверждает необходимость обеспечения целостности процесса выпуска релизов и, соответственно, устанавливает, что теги идентификации программного обеспечения должны полностью соответствовать процессам выпуска релизов, оговоренным в ИСО/МЭК 20000-1:2005. Данная часть настоящего стандарта рекомендует включение дополнительных элементов, относящихся к сведениям о релизах, в теги идентификации программного обеспечения (см. 8.4).

5.4 Согласование с ИСО/МЭК 20000-2:2005 Информационная технология. Управление услугами. Часть 2. Нормы и правила

a) В ИСО/МЭК 20000-2:2005, 10.1.5, оговаривается следующее: «Процессы выпуска релизов и распространения должны разрабатываться и реализовываться таким образом, чтобы a) соблюдалось соответствие стандартам архитектуры систем, стандартам управления услугами и инфраструктурным стандартам провайдера услуг; b) в процессе установки, обработки, упаковки и поставки сохранялась целостность...».

Данная часть настоящего стандарта подтверждает важность обеспечения соответствия процессов выпуска релизов и распространения посредством создания дополнительного элемента «Release package» (пакет релизов) (см. 8.4.17).

b) В ИСО/МЭК 20000-2:2005, 10.1, оговаривается следующее: «В ходе верификации и приемки процессов: a) должна осуществляться проверка того, что управляемое окружение для проведения приемосдаточных испытаний отвечает требованиям целевого производственного окружения; b) должно быть обеспечено создание релиза из версий согласно процессам управления конфигурацией и его установка в окружении для проведения приемосдаточных испытаний с использованием запланированного производственного процесса...».

Данная часть настоящего стандарта подтверждает важность обеспечения соответствия управляемого окружения для проведения приемосдаточных испытаний требованиям целевого производственного окружения посредством создания дополнительного элемента «Release verification» (Верификация релиза) (см. 8.4.19).

c) В ИСО/МЭК 20000-2:2005, 10.1, оговаривается следующее: «Важно, чтобы релиз был поставлен в целости и сохранности в место назначения в состоянии, которое ожидает получатель».

Данная часть настоящего стандарта подтверждает необходимость эффективной и безопасной поставки релизов, предлагая создание дополнительного элемента «Release rollout» (Развертывание релиза), с помощью которого организация сможет проверять, кто именно утвердил программный пакет как готовый к использованию в коммерческих целях, и когда было осуществлено утверждение (см. 8.4.18).

6 Реализация процессов создания и ведения тегов идентификации программного обеспечения

6.1 Общие требования и инструкции

6.1.1 Обзор тегов идентификации программного обеспечения

В приложении А приводится более концептуальный обзор принципов работы с тегами идентификации программного обеспечения (с целью облегчения понимания).

6.1.2 XML и XSD

Тег идентификации программного обеспечения должен определяться как структура XML-данных. Используемым языком XML Schema Definition (Определение схемы XML) (XSD) должен быть язык, определенный в Приложении G, или любая его (обновленная) версия, которую можно загрузить по адресу: gost.e-ecolog.ru.

Также могут иметься дополнительные версии схемы, имеющие идентификатор версии схемы, указанный в пути к схеме. Все предыдущие версии схемы должны сохраняться.

6.1.3 Уникальный регистрационный идентификатор (registration ID, regid)

Дескрипторы идентификации ПО могут создаваться несколькими различными организациями и не обязательно должны регистрироваться через централизованный орган. Кроме того, данная часть настоящего стандарта позволяет организациям создавать дескрипторы идентификации ПО для созданных не ими элементов конфигурации ПО (например, организация может создавать дескрипторы идентификации ПО для собственных внутренних процессов обнаружения программного обеспечения). С целью удовлетворения этих требований в данной части настоящего стандарта используется т. н. regid (регистрационный идентификатор). Идентификатор regid создается на базе классифицированного имени iSCSI, как это определено в стандартах IETF RFC 3720 (см. 3.2.6.3.1) и IETF RFC 3721 (см. 1.1), и представляет собой уникальную ссылку на владельца имен.

Идентификатор regid может быть создан только частным лицом или организацией, являющимися или являвшимися владельцами регистрационных удостоверений на имя домена (как это определено в стандартах IETF RFC 1034 (см. 3.5) и IETF RFC 1123 (см. 2.1)). Имя домена не обязательно должно быть активировано и не обязательно должно преобразовываться в конкретный адрес. Сами по себе имена домена не являются уникальными идентификаторами, поскольку срок их действия может закончиться, и/или они могут быть приобретены другими объектами. Другими словами, в идентификаторе regid также должны содержаться сведения о дате, в которую объект зарегистрировал домен. Для объектов, которые хотят получить еще большую гибкость в присвоении уникальных имен своим подобъектам, в идентификаторе regid выделяется дополнительный суффикс, который можно использовать, например, для того чтобы предоставить крупным разработчикам программных продуктов средства, с помощью которых каждое из их бизнес-подразделений могло бы независимо управлять собственными тегами идентификации программного обеспечения.

Имя идентификатора regid должно состоять из следующих элементов:

– строка «regid» – определяет элемент в качестве регистрационного идентификатора (registration id) для тегов идентификации программного обеспечения;

– символ точки '.';

– код даты в формате ГГГГ-ММ. Этой датой должна быть дата, во время которой объект, присваивающий имя, зарегистрировал домен. Под этой датой должно подразумеваться гринвичское время

00:01 первого дня месяца, в котором объект, присваивающий имя, зарегистрировал на себя домен. В коде даты используется григорианский календарь, и, соответственно, этот код должен включать в себя все четыре цифры года и обе цифры месяца (где январь соответствует 01, а декабрь – 12). Цифры года и месяца должны разделяться символом черточки;

– символ точки '.';

– зарезервированное имя домена объекта, присваивающего имя (физического лица или организации), создающего тег идентификации программного обеспечения;

– необязательная строка, определяющая подобъекты, которые сами могут быть уникальными владельцами имен. Эта строка предваряется следующим символом:

- символ запятой ',',

- владелец имени домена может по своему усмотрению задавать текст, следующий за зарезервированным именем домена (однако без запятых, которые имели бы смысл префикса), при условии, что все символы, присутствующие в тексте, могут использоваться при создании имен файлов на любых платформах, на которых будет устанавливаться тег. Объект, присваивающий имя, должен обеспечить уникальность каждой ссылки на подобъект в пределах организации.

Примеры идентификаторов regid, созданных объектами, владеющими доменами example.com или example.net, могут выглядеть следующим образом:

Таблица 1 – Примеры значений regid

Naming Type	Additional Date	Auth	«example.com» naming authority
regid.	1995-09.	com.example,	AccountingSystems
regid.	1995-09.	com.example	
regid.	1995-09.	net.example,	WordProcessing

6.1.4 Расширение файла тега идентификации программного обеспечения и его размещение для установки

Имена файлов тегов идентификации программного обеспечения должны включать файловое расширение «.swidtag», чтобы эти файлы могли быть распознаны платформами (см. 4.1.7) и инструментарием для обнаружения тегов.

Каждый провайдер платформы (см. 4.1.18), например, поставщик операционной системы, должен указать, в каких местах должны размещаться теги идентификации программного обеспечения. Например, при установке операционной системы Windows® размещение может определяться значением инструментария управления Windows (Windows Management Instrumentation™); дистрибутивы Linux™ могут использовать значение диспетчера пакетов Red Hat (Red Hat Package Manager). Если провайдер платформы укажет каталог размещения, то тег идентификации программного обеспечения должен быть установлен в этом каталоге.

Если провайдер платформы не укажет каталог размещения, теги идентификации программного обеспечения должны быть установлены в общеизвестных каталогах с совместным доступом, содержащих часто используемую системную информацию. В следующих примерах приводится информация о том, какие каталоги с совместным доступом могут использоваться для различных платформ (если провайдер платформы не предоставил собственные спецификации).

Таблица 2 – Примеры размещения файлов тегов на разных платформах

Apple Macintosh™ OS:X™ Leopard	<root>/Library/Application Support/<software creator regid>
Apple Macintosh™ OS X™ pre-Leopard Примечание – Теги идентификации программного обеспечения должны включаться в каталог приложений по умолчанию на всех операционных системах (см. ниже). В системах Pre-Leopard OS X в качестве каталога по умолчанию также должно использоваться это расположение	Application Directory/<program.app package>/contents

Окончание таблицы 2

UNIX® и Linux™	usr/share/<software creator regid>
Windows® NT	C:\Winnt\All Users\Application Data\<software creator regid>
Windows® 2000 Professional Windows Server® 2000 Windows® XP Windows Server® 2003	% AllUsersProfile%\Application Data\<software creator regid>
Microsoft Vista® Microsoft Server® 2008	% Program Data%\<software creator regid>

Провайдер платформы может предоставить доступ к тегу идентификации программного обеспечения с помощью методов, не зависящих от способов доступа к файлам. Например, в ОС Microsoft Vista® включены 4 прикладных программных интерфейса (API), которые могут использоваться для управления хранилищем тегов идентификации программного обеспечения. Такими API-интерфейсами являются следующие:

Таблица 3 – API-интерфейс Microsoft Vista® для управления тегами идентификации программного обеспечения

SLGetInstalledSAMLicenseApplications	Извлекает список приложений, содержащих тег идентификации программного обеспечения, установленный с помощью API SLInstallSAMLicense
SLGetSAMLicense	Считывает информацию о конкретном теге идентификации программного обеспечения, установленном с помощью API SLInstallSAMLicense
SLUninstallSAMLicense	Удаляет тег идентификации программного обеспечения для указанного приложения
SLInstallSAMLicense	Добавляет тег идентификации программного обеспечения в хранилище Microsoft Vista®

Кроме того, копия тега идентификации программного обеспечения также будет установлена в каталог верхнего уровня самого приложения. Таким образом, тег идентификации программного обеспечения может быть обнаружен, даже если съемное устройство хранения данных (например, жесткий USB-диск) будет перенесено с одной системы на другую (см. 7.5). В том случае, если одно и то же программное обеспечение должно быть установлено в двух различных местах для одного и того же набора конечных пользователей, предполагается, что по-прежнему будут иметься два (или более) экземпляров тега идентификации программного обеспечения в указанном выше общем системном каталоге, а также два (или более) других тега идентификации программного обеспечения, располагающихся в корневых каталогах установочных каталогов. В этом случае утилите удаления программного обеспечения в момент удаления должно быть известно о наличии нескольких установок, чтобы не удалить тег идентификации программного обеспечения из общего системного каталога, пока все другие установки не будут удалены.

Примечание – Поставщики инструментария SAM должны помнить о том, что могут встречаться случаи, когда тег идентификации программного обеспечения может присутствовать как в общем системном каталоге, так и в одном или нескольких каталогах верхнего уровня места установки программного пакета. Используя информацию, содержащуюся в элементе installation_details, инструментарий SAM может сопоставлять теги, расположенные в общем системном каталоге и корневых установочных каталогах, с их соответствующими установками. В инструментарий SAM должны быть включены правила, позволяющие учитывать различные перестановки обнаруженных тегов. В тех случаях, когда тег обнаруживается в каталоге установки программного пакета, а не в общем системном каталоге, то для того чтобы определить, располагается ли программное обеспечение на съемном устройстве хранения данных (которое, возможно, было перенесено на другую систему), требуется применение дополнительных правил, . Если тег обнаруживается в общем системном каталоге, и, кроме того, несколько тегов обнаруживаются в установочных каталогах, может потребоваться применение правил, обеспечивающих соответствие организационным политикам. Специалист-практик по использованию процессов SAM может в этом случае оформить соответствующий отчет и предпринять соответствующие действия.

Целью инструментария SAM должно быть максимальное упрощение для специалиста-практика по использованию процессов SAM процессов управления исключениями из организационной политики, выявление таких исключительных случаев и выдача отчетов по ним в соответствии с политиками, указанными специалистом-практиком, и значительное упрощение процессов контроля над общим ходом реализации процессов SAM.

6.1.5 Уникальные идентификаторы

С целью обеспечения уникальности определяются два элемента, совместно создающие глобальный уникальный идентификатор (ID), называемый software_id. Это два следующих элемента:

a) tag_creator_regid;

b) unique_id, который может являться либо идентификатором GUID, либо любой ссылкой, уникальной для элемента tag_creator_regid. Элемент unique_id должен удовлетворять ограничениям на использование символов в URI, как это определено в стандарте IETF RFC 3986, раздел 2, Символы.

Преимущества использования уникального идентификатора в процессе создания тега идентификации программного обеспечения, в частности, состоят в следующем:

a) идентификация отношений родительский-дочерний объект;

b) явное определение зависимостей и выявление зависимого программного обеспечения;

c) идентификация обновленного программного обеспечения и разрешенных пакетов обновлений;

d) возможность ссылаться на идентифицирующие теги идентификации программного обеспечения из элементов конфигурации программного обеспечения.

6.1.6 Уникальное имя файла тега идентификации программного обеспечения – распространение

При создании тега идентификации программного обеспечения для его распространения на установочный носитель выдача дополнительных уникальных id, присущих конкретной системе, как это описано в 6.1.7, невозможна, поскольку на данный момент времени тег пока еще не был установлен ни на одно вычислительное устройство. В этом случае файл тега является частью эталонного образа для процесса установки и должен соответствовать следующей структуре:

<tag_creator_regid>_<software_id.unique_id>.swidtag

В соответствии с данной структурой устанавливается уникальное имя файла, который может сопровождать поставляемое программное обеспечение.

Таким образом, файл тега идентификации программного обеспечения на установочном носителе в общем случае может иметь следующее имя:

regid.1986-12.com.adobe_fc3cc419-b5a1-9f16-ed203e537c40.swidtag

После установки тега процедура установки выполняется в соответствии процессом, описанным в 6.1.7.

6.1.7 Уникальное имя файла тега идентификации программного обеспечения – установлено

После того как программное обеспечение установлено на вычислительное устройство, имя файла тега идентификации программного обеспечения будет уникальным по крайней мере для данного устанавливаемого кода и будет соответствовать следующей структуре:

<tag_creator_regid>_<software_id.unique_id>_<unique_sequence_id>.swidtag

Элемент элемент unique_sequence_id является необязательным и должен использоваться для обеспечения уникальности имени файла каждого тега идентификации программного обеспечения, установленного на вычислительном устройстве. Элемент unique_sequence_id может быть простой числовой последовательностью, или его можно создать с помощью алгоритма, выбранного организацией, устанавливающей тег идентификации программного обеспечения. Метод создания элемента unique_sequence_id выбирает организация, устанавливающая тег идентификации программного обеспечения, однако используемая методология должна обеспечивать уникальность имени файла для конкретного машинного окружения и/или виртуальной среды. Начальная часть имени файла тега идентификации программного обеспечения должна выглядеть следующим образом:

regid.1986-12.com.adobe_fc3cc419-b5a1-9f16-ed203e537c40

За ней должен следовать элемент unique_sequence_id. Алгоритм, определяющий элемент unique_sequence_id, может использовать несколько условий:

1. Для имен файлов базовых тегов элемент unique_sequence_id может быть простым порядковым номером. Если в процессе установки тега идентификации программного обеспечения выяснится, что тег с таким именем уже существует, порядковый номер элемента увеличивается на единицу. В этом случае числовая последовательность должна начинаться с определенного ссылочного номера и увеличиваться на единицу по мере установки других тегов идентификации программного обеспечения для одного и того же ID программного обеспечения (это может происходить тогда и только тогда, когда про-

граммное обеспечение допускает несколько установок на конкретном вычислительном устройстве). Обращаем внимание, что элемент installation_instance, содержащийся внутри дополнительного элемента installation_details, может описанным выше способом использоваться в качестве части имени файла.

Пример имени файла тега идентификации программного обеспечения (для первой установки):

regid.1986-12.com.adobe_fc3cc419-b5a1-9f16-ed203e537c40_1.swidtag

tag_creator_regid unique_id Порядковый номер

2. Предпочтительный алгоритм для элемента unique_sequence_id должен учитывать данные, непосредственно относящиеся к вычислительному устройству и/или носителю, на которые устанавливается программное обеспечение (такими данными могут быть серийный номер устройства, MAC-адреса сетевых карт, серийный номер жесткого диска или иные уникальные ссылочные номера) и/или данные о виртуальной среде, в которой было установлено программное обеспечение. Обращаем внимание, что элемент installation_target_id, содержащийся внутри дополнительного элемента installation_details, может описанным выше способом использоваться в качестве части имени файла. Наличие данных, привязанных к устройству, позволяет упростить процессы отслеживания тегов идентификации программного обеспечения, которые могут быть установлены на съемных или совместно используемых носителях. Обратите внимание, что данный алгоритм может также потребовать использования уникального порядкового номера для гарантии того, что несколько установок на одной и той же машине не создают одно и то же имя файла. Элемент installation_instance, содержащийся внутри дополнительного элемента installation_details, может описанным выше способом использоваться в качестве части имени файла. Пример имени файла тега идентификации программного обеспечения:

regid.1986-12.com.adobe_fc3cc419-b5a1-9f16-ed203e537c40_001c26f781fb_1.swidtag

tag_creator_regid unique_id installation_target_id Порядковый номер

Независимо от выбранного алгоритма имя файла тега идентификации программного обеспечения должно создаваться процедурой установки, устанавливающей тег, и это имя не должно конфликтовать ни с каким существующим именем файла. Кроме того, количество символов, используемых в имени файла, не должно превышать 254 (или меньшее количество, если целевая платформа для программного обеспечения требует использования более коротких имен файлов). И, наконец, символы, используемые в имени файла, должны отвечать всем специфическим требованиям, предъявляемым файловыми системами, на которых будет устанавливаться тег.

Файловое расширение .swidtag должно использоваться для всех тегов идентификации программного обеспечения. Данная схема именования позволяет применять к одному наименованию продукта несколько тегов идентификации программного обеспечения, обеспечивая, таким образом, поддержку обновлений. Данная схема также позволяет создавать уникальные теги идентификации программного обеспечения организациями, а не создателем оригинального программного обеспечения, что можно использовать для определения тегов для устаревшего программного обеспечения.

6.1.8 Согласованность значений данных

Одной из проблем, с которыми можно столкнуться при управлении программными активами, является то обстоятельство, что торговые наименования, используемые поставщиками программного обеспечения, часто меняются. Поскольку тег идентификации программного обеспечения обычно невидим для конечного пользователя, данный тег может служить согласованным и надежным источником информации, на основании которой можно осуществлять эффективное управление инвентаризацией и выверку прав на использование программного обеспечения. Возможность сохранения согласованности значений для разных версий программного обеспечения или для продуктов одной линейки – одно из главных преимуществ, которые предоставляют теги идентификации программного обеспечения для специалиста-практика по использованию процессов SAM.

Многочисленные элементы, содержащиеся в теге идентификации программного обеспечения, должны оставаться согласованными от тега к тегу, чтобы можно было оптимизировать весь процесс управления программными активами. Рекомендуется, например, чтобы поставщики тегов сохраняли согласованность следующих элементов:

а) идентификационные данные создателя программного обеспечения (см. 8.3.4) – этот элемент должен оставаться согласованным по всем программным пакетам, созданным конкретной компанией;

b) идентификационные данные лицензиара программного обеспечения (см. 8.3.5) – этот элемент должен оставаться согласованным по всем программным пакетам, лицензированным конкретной компанией;

c) идентификационные данные создателя тега (см. 8.3.7) – этот элемент должен оставаться согласованным по всем тегам идентификации программного обеспечения, созданным конкретной компанией;

d) информация о лицензии и канале (см. 8.4.8) – структура этого дополнительного элемента должна оставаться согласованной по всем линейкам продуктов; рекомендуется сохранять согласованность значений по всему программному обеспечению, созданному конкретной компанией. Обратите внимание, что данные элементов «license and channel information» (информация о лицензии и канале) тега идентификации программного обеспечения определяют не лицензию на программное обеспечение или права на использование программного обеспечения, а инструкции для специалистов-практиков по использованию процессов SAM, которые помогут им определить и, возможно, автоматизировать процедуры выверки лицензий на программное обеспечение;

e) категория продукта (см. 8.4.12) – этот дополнительный элемент должен оставаться согласованным для конкретного продукта до тех пор, пока функциональность продукта не будет увеличена или уменьшена до такой степени, что станет очевидна его принадлежность к другой категории;

f) идентификатор продукта (см. 8.4.14) – этот дополнительный элемент должен оставаться согласованным для продуктов, в отношении которых имеются соглашения об обслуживании, предоставляющие права на получение обновлений в течение периода времени, когда данный элемент используется для идентификации конкретного продукта разных релизов – этот элемент представляет собой не наименование продукта, а простой идентификатор, поэтому установление отношений для целей обновления может выполняться автоматически.

6.1.9 Обнаружение тегов идентификации программного обеспечения

Программное обеспечение в общем случае создается поставщиками программного обеспечения в виде исходной эталонной копии, затем копируется и распространяется по различным каналам. В зависимости от требований поставщика программного обеспечения, тег идентификации программного обеспечения может быть внедрен непосредственно в исходную эталонную копию или же может быть создан программой-установщиком или даже самим программным пакетом. Основным требованием является то, чтобы тег идентификации программного обеспечения для пакета можно было обнаружить в машинном окружении, окружении носителя и/или в виртуальной среде, на которых устанавливается пакет. Более подробные сведения приведены в приложении с описанием инструкций для поставщиков программного обеспечения (приложение B).

6.1.10 Языки

Признавая тот факт, что многие создатели программного обеспечения производят программное обеспечение с разными сборками, привязанными к языку, а многие другие производят программное обеспечение только с одной сборкой, реализующей дополнительные «языковые пакеты», данная часть настоящего стандарта не требует, чтобы теги идентификации программного обеспечения могли распознавать различные языковые версии одного и того же продукта. Тем не менее, настоятельно рекомендуется обеспечить согласованную категоризацию посредством использования элемента «supported languages» (поддерживаемые языки) (см. 8.4.24).

Предлагаемой методологией кодирования тегов идентификации программного обеспечения, создаваемых на базе данной части настоящего стандарта, является utf-8 (см. gost.e-ecolog.ru
O-charset).

6.1.11 Владение элементами, содержащимися в тегах идентификации программного обеспечения

Учитывая способы, которыми различные объекты могут добавлять или изменять элементы тега идентификации программного обеспечения, для каждого из этих объектов может быть необходимо знать, какие именно элементы создал и/или изменил этот объект. Данная возможность обеспечивается, в основном, с помощью элемента elements_owner посредством использования внутреннего ID элемента, как это описано в 6.1.12.

Кроме того, владение отдельными элементами может устанавливаться посредством использования цифровых подписей, как это описано в 6.1.13. Стоит отметить, что использование цифровых подписей не определяет владельца непосредственно, как это делает элемент elements_owner, поэтому в любом случае, даже при использовании цифровых подписей, рекомендуется использовать элемент elements_owner.

6.1.12 Внутренний ID элемента

Для элементов, содержащихся в теге идентификации программного обеспечения, должна быть предусмотрена возможность делать перекрестные ссылки на другие отдельные элементы, содержащиеся в теге идентификации программного обеспечения. Одним из очевидных примеров, когда это может потребоваться, является необходимость идентификации элементов, которые были созданы и/или изменены конкретным создателем или модификатором тега, указанным в элементе elements_owner. Такая возможность делать перекрестные ссылки реализуется посредством назначения внутренних идентификаторов (ID) элемента отдельным элементам. Такие идентификаторы затем могут использоваться для перехода по перекрестной ссылке на эти элементы из других элементов.

Внутренние ID элемента используются для переходов по ссылке внутри тега, а также для переходов по ссылке между тегами, когда необходимо идентифицировать конкретный элемент другого тега (это может понадобиться в том случае, если во вторичном файле присутствует цифровая подпись). Такие ID элемента должны быть уникальными в теге идентификации программного обеспечения, но могут быть не уникальными для разных тегов. Атрибуты ID в тегах идентификации программного обеспечения обычно используются для идентификации конкретного владельца элементов. Для определения и использования атрибутов ID создатель и модификатор тега могут использовать любую определенную методологию. Однако в большинстве случае рекомендуется использовать следующие правила по умолчанию, установленные в данной части настоящего стандарта:

a) все XML ID должны начинаться с буквенного символа. Согласно правилам данной части настоящего стандарта, ID должен начинаться с буквы «e», что означает «element» (элемент);

b) все элементы верхнего уровня должны использовать то же значение, что и значение пункта, определяющего данный элемент, причем каждый компонент пункта должен обозначаться символом подчеркивания. Элемент product_version, таким образом, получает следующий атрибут:

<swid:product_version ID="e8_3_3">;

c) любой элемент может иметь подэлементы. В этих случаях ID каждого подэлемента должен начинаться с идентификатора пункта верхнего уровня (как это описано выше), за которым следует конструкция «sub [номер подэлемента]». Для структуры элемента product_version, таким образом, определяются следующие ID:

<swid:name ID="e8_3_3sub1">10.2</swid:name>
<swid:numeric ID="e8_3_3sub2">
<swid:major ID="e8_3_3sub2sub1">10</swid:major>
<swid:minor ID="e8_3_3sub2sub2">2</swid:minor>
<swid:build ID="e8_3_3sub2sub3">0</swid:build>
<swid:review ID="e8_3_3sub2sub4">0</swid:review>
</swid:numeric>

d) для значений, которые могут иметь несколько записей, например, аннотаций, ID должен начинаться с идентификатора пункта верхнего уровня (как это описано выше), за которым следует порядковый номер, определенный в «seq [номер экземпляра]». Для структуры аннотации, таким образом, определяются следующие ID:

<swid:abstract lang=«en» ID="e8_4_1_seq1">Это аннотация на английском языке</swid:abstract>
<swid:abstract lang=«fr» ID="e8_4_1_seq2">Это аннотация на французском языке</swid:abstract>

Следование данным рекомендациям для ссылок по ID, используемых в элементе elements_owner, позволяет реализовать согласованный подход к определению ID, которые могут быть легко поняты специалистом, изучающим тег, а также осуществлять эффективное управление посредством использования автоматизированных подходов.

6.1.13 Аутентичность тегов идентификации программного обеспечения

Часто бывает необходимо подтвердить аутентичность тега идентификации программного обеспечения. Например, в процессе аудита поставщик программного обеспечения может пожелать убедиться в том, что в теге идентификации программного обеспечения, полученном в процессе обнаружения,

отсутствуют измененные основные элементы тега. Аутентификация обеспечивается за счет использования цифровых подписей внутри тега идентификации программного обеспечения.

Консорциум W3C опубликовал действующую рекомендацию, определяющую необходимость выдачи цифровых подписей в документе XML. Данная рекомендация называется «XML-Signature Syntax and Processing (Синтаксис и обработка XML-подписи) (Второе издание) – 10 июня 2008 г.» и расположена по следующему адресу:

http://www.w3.org/TR/xmldsig-core/

Примечание – W3C не называет справочные официальные документы «стандартами». Официально поддерживаемые документы носят название «рекомендации». Читатель должен помнить о том, что для вывода документа на рекомендательный уровень консорциум W3C использует тщательным образом проработанные процессы, и что рекомендации W3C должны расцениваться как официальные документы.

В рекомендации W3C определяются сервисы аутентификации сообщений с защитой целостности, а также сервисы аутентификации подписывающего лица для данных любого типа.

Данная часть настоящего стандарта не оговаривает процессы применения цифровых подписей к тегам идентификации программного обеспечения, поскольку на этот счет существует рекомендация W3C.

Подписи не являются обязательной частью тега идентификации программного обеспечения и могут использоваться, если этого потребует любой создатель или модификатор тега, для того чтобы гарантировать неизменность разделов тега и/или обеспечить аутентификацию подписывающего лица. Если для тега идентификации программного обеспечения потребуются подписи, такие подписи должны соответствовать рекомендации консорциума W3C, определяющей синтаксис XML-подписи (http://www.w3.org/TR/xmldsig-core/).

Примечание – Теги идентификации программного обеспечения, в общем случае, не требуют применения усовершенствованных электронных XML-подписей (Advanced Electronic Signatures, XAdes), поэтому в данной части настоящего стандарта отсутствуют ссылки на данную рекомендацию W3C.

6.1.14 Стандартизация определения XSD

В теге идентификации программного обеспечения используются несколько стандартизованных типов, в том числе специальные записи даты/времени, которые должны быть оформлены в соответствии форматами, определенными в рекомендации W3C «XML Schema Part 2: Datatypes (Схема XML Часть 2: Типы данных) (Второе издание), 2004 г. Более подробная информация об этих типах данных приведена по следующему адресу:

gost.e-ecolog.ru

При использовании специальных типов данных, оговоренных в указанной выше рекомендации W3C, к тегам идентификации программного обеспечения можно применять процесс автоматической проверки (валидации), что позволяет обеспечить более согласованную структуру предоставляемых данных.

Примечание – Несколько элементов, определенных в данной части настоящего стандарта, требуют использования регулярных выражений. Синтаксис таких регулярных выражений также определен в указанной выше рекомендации W3C.

6.2 Жизненный цикл работы с тегами идентификации программного обеспечения: порядок операций

6.2.1 Введение

На следующем рисунке приведен жизненный цикл создания тега идентификации программного обеспечения, начиная от создателя программного обеспечения (или создателя тега) и заканчивая организацией потребителя программного обеспечения.

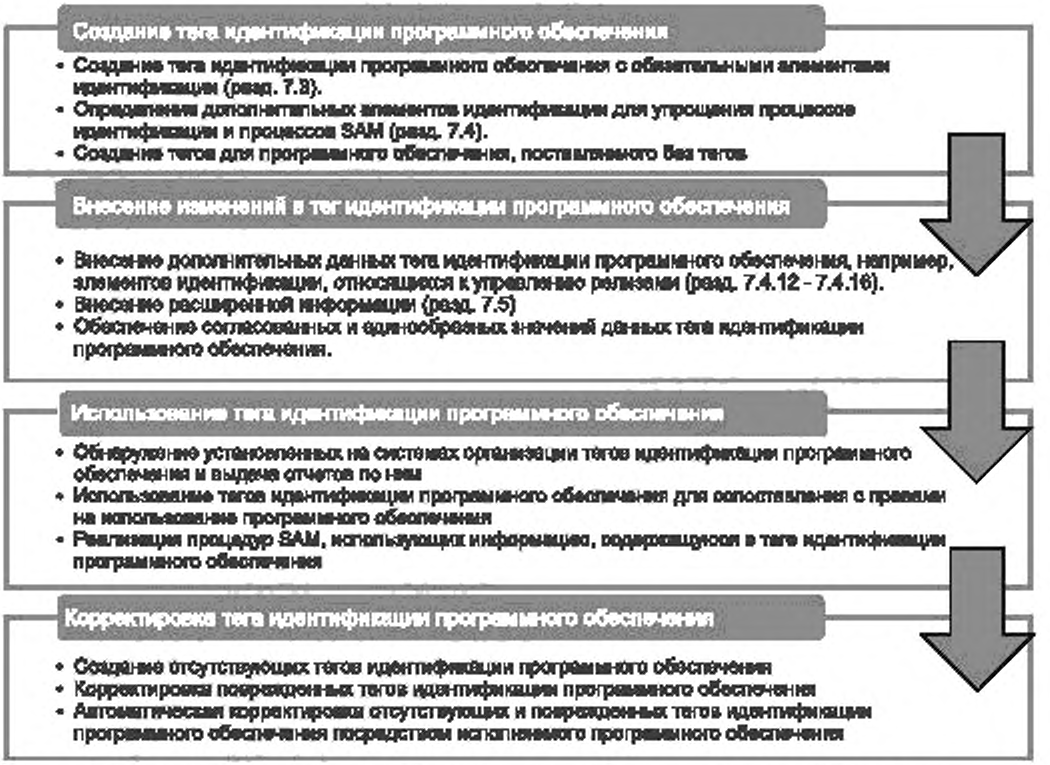

Рисунок 1 – Жизненный цикл работы с тегами идентификации программного обеспечения

6.2.2 Создание тега идентификации программного обеспечения

Создателем тега идентификации программного обеспечения часто является сам создатель программного обеспечения. После создания программного обеспечения создатель программного обеспечения разрабатывает соответствующий тег идентификации программного обеспечения, чтобы идентифицировать программный пакет.

Создателями тегов идентификации программного обеспечения могут быть следующие лица:

а) Производители программного обеспечения

Тег идентификации программного обеспечения может создаваться в процессе разработки производителем программного обеспечения конкретного элемента конфигурации программного обеспечения. Соответственно, тег идентификации программного обеспечения и программа установки программного обеспечения отгружаются вместе. Программное обеспечение может направляться либо непосредственно потребителям программного обеспечения или издателям, либо и тем и другим. Различные стороны могут договариваться между собой о том, кто создает тег идентификации программного обеспечения, и кто определяет источник элемента конфигурации программного обеспечения. Возможно, что этот процесс стороны будут выполнять совместно, так как он зависит от канала, используемого для распространения программного обеспечения. Производители программного обеспечения, по всей вероятности, будут использовать в составе своего программного обеспечения цифровые подписи.

b) Издатели программного обеспечения

В тех случаях когда программное обеспечение разрабатывается одной организацией, а затем издается другой, в процессе упаковки издатель программного обеспечения продуктов создает тег идентификации программного обеспечения, который должен быть включен в процесс установки продукта.

c) Разработчики приложений для направления бизнеса

Разработчики приложений для направления бизнеса также создают теги идентификации программного обеспечения, которые должны быть включены в процесс установки продукта.

d) Дистрибьюторы, переупаковщики, продавцы, создающие добавочную стоимость товара, и другие организации, вносящие изменения в теги

Организации, распространяющие программное обеспечение, в которое не включены стандартизированные теги идентификации программного обеспечения, могут пожелать добавить такие теги с целью удовлетворения требований своих потребителей программного обеспечения. Если программный пакет не включает в себя тег идентификации программного обеспечения, специалистам-практикам по использованию процессов SAM рекомендуется создать и включить в этот пакет собственный тег идентификации программного обеспечения, что позволит оптимизировать процессы SAM.

Создателями тегов идентификации программного обеспечения также могут являться сторонние организации, не связанные непосредственно с создателем программного обеспечения. Сторонние организации могут создавать теги, в частности, для устаревшего программного обеспечения, которое не имеет тегов идентификации программного обеспечения, или для программного обеспечения, разработанного организацией, которая решила не включать теги идентификации программного обеспечения в свои продукты. В этих случаях создатель тега должен определить значения элементов в теге идентификации программного обеспечения, которые указывают, что эти создатели не являются создателем программного обеспечения (например, значение tag_creator_regid в элементе the software_id). Такие случаи считаются очевидными, поскольку значение tag_creator_regid будут отличаться от значения software_creator_regid.

6.2.3 Внесение изменений в тег идентификации программного обеспечения

Физические лица или компании, вносящие изменения в теги идентификации программного обеспечения и/или добавляющие в них вспомогательную информацию, считаются модификаторами тегов. К данной группе лиц могут относиться агрегаторы программного обеспечения, продавцы, создающие добавочную стоимость товара, а также группы внутри организаций, занимающиеся управлением процессами выпуска релизов программного обеспечения.

Модификаторами тегов идентификации программного обеспечения могут быть следующие лица:

a) дистрибьюторы;

b) реселлеры;

c) продавцы, создающие добавочную стоимость товара;

d) издатели, выпускающие переиздания;

e) упаковщики;

f) поставщики инструментария для обнаружения тегов;

g) поставщики инструментария для развертывания;

h) администраторы релизов.

6.2.4 Использование тега идентификации программного обеспечения

Несмотря на то что конечными выгодоприбретателями данных стандартизированных тегов идентификации программного обеспечения являются потребители программного обеспечения, с этими данными работают, в основном, инструментарий для обнаружения тегов и операционные системы. Они считывают данные тега идентификации программного обеспечения, чтобы получить информацию о конкретном наборе элементов конфигурации программного обеспечения. После того как элемент конфигурации программного обеспечения будет размещен на конкретном вычислительном устройстве, надлежащим образом оформленный тег идентификации программного обеспечения обеспечит достоверную идентификацию конкретного установленного программного обеспечения, создателя тега и всех сторонних поставщиков, которые могли вносить изменения в программное обеспечение. Для специалиста-практика по использованию процессов SAM эта информация является бесценной.

Конечными пользователями тегов идентификации программного обеспечения могут быть следующие лица:

a) владельцы процессов SAM;
b) специалисты служб IT-поддержки;
c) владельцы элемента конфигурации программного обеспечения.

6.2.5 Корректировка тега идентификации программного обеспечения

Несмотря на то, что управление тегами идентификации программного обеспечения должно осуществляться вместе с управлением всеми другими компонентами, связанными с конкретной установкой программного обеспечения, теги могут удаляться или повреждаться (случайно или намеренно).

В том случае, если тег идентификации программного обеспечения окажется поврежден, в программном приложении предусмотрены дополнительные методы восстановления действительности тега идентификации программного обеспечения и корректировки и/или, возможно, автоматического самовосстановления тега идентификации программного обеспечения. Эти методы помогают обеспечить более высокий уровень целостности данных, собираемых инструментарием для обнаружения тегов.

В том случае, если установка продукта не включает в себя тег идентификации программного обеспечения, администраторы релизов могут, с целью упрощения процессов выверки SAM, создать собственный тег идентификации программного обеспечения для включения в программное обеспечение.

Во всех случаях должны быть предусмотрены методы, с помощью которых средства или сервисы обнаружения могли бы выполнять перекрестный контроль информации способом, обеспечивающим гарантированную точность данных. Например, агент обнаружения может собирать теги идентификации программного обеспечения и все имена исполняемых файлов в системе. В тегах идентификации программного обеспечения предусмотрена возможность указания всех файлов, связанных с конкретной частью программного обеспечения. С помощью соответствующего алгоритма инструментарий SAM для обнаружения тегов может быстро убедиться в том, что система с конкретным тегом идентификации программного обеспечения также включает в себя необходимые файлы, связанные с этим программным обеспечением. После этого механизм выверки может отфильтровать из собранного списка имен исполняемых файлов известные имена файлов приложений. Любые имена файлов, оставшиеся в списке после завершения данного процесса, будут считаться исключениями, и специалисту-практику по использованию процессов SAM, возможно, нужно будет внимательнее присмотреться к этим файлам.

7 Требования и инструкции по использованию платформ

7.1 Типы платформ

В контексте данной части настоящего стандарта под термином «платформа» понимается компьютерное оборудование, или аппаратное устройство, и/или соответствующая операционная система, или виртуальная среда, на которых может устанавливаться или запускаться программное обеспечение (см. 4.1.17). Операционная система Linux™, например, используется на множестве устройств, начиная от сотовых телефонов и заканчивая универсальными электронными вычислительными машинами, и в контексте данной части настоящего стандарта каждый вариант такого использования можно считать отдельной платформой. Другие примеры платформ (список может быть продолжен):

a) redhat™ Linux™ Enterprise 4.0 Intel x86;
b) novell SuSE™ Linux™ 10.2 Intel x64;
c) macintosh™ OS 10.4 Intel x64;
d) microsoft Vista® x64;
e) HP-UX 11i Itanium™

Платформа также может быть реализована в виртуальной среде (например, Java™ или .NET™), и в этих случаях вопросы аппаратной поддержки, в общем случае, не важны. Большое значение в виртуальных средах, тем не менее, имеет контроль версий. Программное обеспечение, написанное и скомпилированное для одной версии Java™ или .NET™, например, может не запускаться на предыдущих версиях виртуальной среды. Предполагается, что виртуальные среды также будут предоставлять тег идентификации программного обеспечения, чтобы идентифицировать конкретные данные по версии установленного окружения.

Примечание – Не следует путать термины «виртуальная среда» и «виртуальная машина». Виртуальная машина может запускаться на главной платформе операционной системы, но оставаться, тем не менее, замкнутой операционной средой. Виртуальная машина, поставляемая вместе с виртуальным оборудованием, соответственно, должна расцениваться как отдельный экземпляр оборудования, как и любая другая отдельная физическая машина.

7.2 Базовые сервисы платформы

Платформы существуют независимо от данных тегов идентификации программного обеспечения содержащихся на платформах элементов конфигурации программного обеспечения, и должны быть безразличны к ним. Тем не менее, на платформе должны быть определены процессы для эффективного хранения и извлечения таких тегов идентификации программного обеспечения.

Рекомендуется, чтобы платформы хранили и извлекали теги идентификации программного обеспечения с использованием процесса, аналогичного тому, который применяют операционные системы при работе с файлами, за тем исключением, что теги идентификации программного обеспечения, с целью упрощения их обнаружения, должны сохраняться в централизованном расположении. Если для хранения тегов идентификации программного обеспечения централизованное хранилище не предусмотрено, они должны сохраняться в общем каталоге, связанном с определяемыми этими тегами элементами конфигурации программного обеспечения (как это определено в 6.1.4), а также каталоге верхнего уровня каждой установки программного пакета. Другими словами, для того чтобы выполнить полную инвентаризацию тегов, инструментарий для обнаружения тегов должен собирать теги идентификации программного обеспечения, которые могут размещаться в нескольких каталогах (например, в каталоге верхнего уровня установленных файлов программного пакета).

Платформа, отвечающая требованиям данной части настоящего стандарта, должна реализовывать следующие сервисы:

a) базовые операции добавления, изменения, считывания и удаления;

b) функции проверки:

1) идентификация того, кто именно установил данный элемент конфигурации программного обеспечения, и когда была выполнена установка,

2) идентификация того, кто именно изменил данный элемент конфигурации программного обеспечения, и когда было выполнено изменение,

3) Идентификация того, кто именно удалил данный элемент конфигурации программного обеспечения, и когда было выполнено удаление.

Примечание – После удаления программного обеспечения теги идентификации программного обеспечения можно не сохранять. С целью обеспечения целостности эти теги рекомендуется удалять. Вместо сохранения тегов ведутся журналы регистрации, по которым, если будет необходимо, можно идентифицировать предыдущие установки;

c) безопасность:

1) определение того, кому разрешается создавать и изменять теги идентификации программного обеспечения,

2) Определение того, кому разрешается считывать теги идентификации программного обеспечения.

7.3 Виртуальные среды

Виртуальные среды обычно устанавливаются на вычислительное устройство и должны предоставлять собственные теги идентификации программного обеспечения, чтобы идентифицировать себя для инструментария для обнаружения тегов.

Пример – Если вычислительное устройство установлена среда Java™ Virtual Machine (JVM), установка должна осуществляться с тегами идентификации программного обеспечения точно так же, как при установке любого другого программного пакета.

После этого другие пакеты могут использовать данные глобального уникального идентификатора и другую идентификационную информацию для тега идентификации программного обеспечения виртуальной среды с целью проверки совместимости элемента конфигурации программного обеспечения с виртуальной средой.

Пример – После установки виртуальной машины JVM к идентификационной информации из этого пакета могут обращаться приложения на базе Java™, использующие JVM. Определять отношения зависимости с JVM должны приложения на базе Java™.

7.4 Виртуальные машины

Виртуальные машины реализуют гостевое операционное окружение, не зависящее от главного операционного окружения. Поскольку виртуальные машины имеют дело с программным обеспечением, процессы установки и обнаружения практически не отличаются друг от друга, за тем исключением,

что процесс обнаружения выполняется полностью на виртуальной машине или на диске виртуальных машин (зачастую это обычный файл, размещающийся в главной операционной системе). В этих окружениях должен предоставляться тег идентификации программного обеспечения точно так же, как для любого другого компьютера.

Предполагается, что в рамках процесса обнаружения инструментарий для обнаружения тегов будет собирать информацию о системе, на которой выполняется обнаружение тега. Если система представляет собой виртуальную машину, также должны собираться сведения об этой виртуальной машине, ее хостинговой среде и пр. После этого процесс выверки SAM использует сведения о собранных тегах идентификации программного обеспечения, а также сведения о среде, в которой установлен тег (тип виртуальной машины, хостинговая среда для виртуальной машины и пр.) для выполнения выверки.

В настоящее время используется множество технологий виртуализации, поэтому в данной части настоящего стандарта не могут быть приведены описания процессов использования тегов идентификации программного обеспечения для всех существующих или будущих технологий. Поставщики средств виртуализации, тем не менее, должны иметь в виду, что с целью обеспечения соблюдения прав на использование программного обеспечения необходимо контролировать и отслеживать процессы установки и использования приложений. Таким образом, такие технологии виртуализации должны предоставлять средства обнаружения программных пакетов или приложений, доступных для использования, и/или используемых на конкретном вычислительном устройстве. Это можно обеспечить посредством предоставления тегов идентификации программного обеспечения, обнаруживаемых во всей виртуализированной среде, или которые могут предоставляться посредством запуска процесса обнаружения на виртуализированном диске.

7.5 Поддержка программного обеспечения, установленного на съемном носителе

Программное обеспечение часто может устанавливаться на съемный носитель. В этих случаях тег идентификации программного обеспечения должен идентифицировать установку программного обеспечения на конкретное вычислительное устройство, а также предоставлять информацию, которая отслеживала бы перемещение съемного носителя. Это осуществляется посредством занесения тега идентификации программного обеспечения в общий системный каталог (либо в хранилище тегов, определенное операционной системой, либо в общие каталоги) вычислительного устройства. Вторая копия тега идентификации программного обеспечения должна быть занесена в каталог верхнего уровня, используемый для установки программного пакета.

Поставщики инструментария должны уметь распознавать случаи, когда на одном вычислительном устройстве может быть обнаружено два или более тегов идентификации, и уметь распознавать случаи, когда программный пакет установлен на вычислительном устройстве и только на нем. В том случае, если съемный носитель обнаруживается на другом вычислительном устройстве, в схеме определения прав на использование программного обеспечения для этого конкретного пакета должно быть указано, должен ли пакет считаться одним объектом (установка на съемном носителе) или двумя объектами (система, на которую было установлено программное обеспечение, а также фактическое программное обеспечение на съемном носителе).

7.6 Идентификация оборудования и платформы

Данная часть настоящего стандарта, в основном, посвящена тегам идентификации программного обеспечения. В большинстве сценариев использования сбор таких тегов идентификации программного обеспечения осуществляется в рамках выполнения стандартных процессов SAM или процессов обнаружения/инвентаризации. Предполагается, что инструмент, используемый в процессе обнаружения, будет собирать и возвращать информацию об оборудовании и другую информацию о платформе (например, сведения об операционной системе), которые будут сопоставляться с любыми собранными тегами идентификации программного обеспечения. Таким образом, будет происходить автоматическое сопоставление программного обеспечения с оборудованием и платформой, на котором оно установлено.

По мере того как портативные устройства будут становиться все более функциональными, виртуальные среды и виртуальные машины будут становиться все более разнообразными, а процессы автоматизированного сбора данных будут становиться все более совершенными, предполагается, что для нормального выполнения процессов управления IT-активами может потребоваться дополнительная идентификационная информация об оборудовании и платформе. Если будет определен официальный

стандарт или стандарт де-факто, определяющий дополнительную уникальную идентификационную информацию об оборудовании и платформе, то для разработки согласованного элемента installation_target_id, который можно будет использовать на любой платформе, должны будут использоваться оговоренные в этом стандарте данные. Если некоторые типы идентификационной информации об оборудовании станут официальным стандартом, предполагается, что информация из этого стандарта будет включена в раздел соответствия будущей версии данной части настоящего стандарта.

8 Элементы

8.1 Общие положения

Элементы описывают общие атрибуты всех или большинства элементов конфигурации программного обеспечения. Операционные системы и инструментарий для обнаружения тегов могут использовать эти атрибуты для идентификации элементов конфигурации программного обеспечения.

Элементы могут изменяться различными модификаторами тегов (см. 6.2.3, 6.2.4).

В данной части настоящего стандарта перечислены 37 элементов (список элементов не полный). Эти элементы являются предопределенными, чтобы обеспечить согласованность между тегами идентификации программного обеспечения (см. 8.2).

Описания элементов, приведенные в 8.3 и 8.4, соответственно, сопровождаются примерами. Примеры приводятся в XML-синтаксисе, формат которого должен использоваться при создании тегов идентификации программного обеспечения. Примеры показывают, какая информация должна быть включена в элементы данных. Расширенные примеры тегов идентификации программного обеспечения приведены в приложении Н.

Данная часть настоящего стандарта не требует наличия конкретного процесса формирования содержания элементов.

Обязательные элементы (см. 8.3) требуются для признания тега идентификации программного обеспечения действительным или полным. Инструментарий для обнаружения тегов должен распознавать неполные теги идентификации программного обеспечения как недействительные и уведомлять специалиста-практика по использованию процессов SAM, отвечающего за теги идентификации программного обеспечения, о том, что эти теги не являются действительными или полными. Всего имеется пять обязательных элементов.

Рекомендуется, чтобы поставщики тегов имели централизованное хранилище всех тегов идентификации программного обеспечения, созданных для всех релизов продуктов, в котором хранились бы, по крайней мере, обязательные элементы. Такое хранилище можно будет использовать для проверки уникальности идентификаторов GUID, а также для обеспечения того, что имя создателя программного обеспечения и идентификатор GUID остаются согласованными по всей линейке продуктов. Данная часть настоящего стандарта не требует привлечения внешнего регистрирующего органа для работы с тегами идентификации программного обеспечения, поэтому уникальность или неуникальность конкретного тега определяет сам создатель тега.

Дополнительные элементы (см. 8.4) могут быть, а могут не быть представлены в теге идентификации программного обеспечения. Элементы данных, соответствующие дополнительным элементам, наделяют создателей тегов идентификации программного обеспечения дополнительной функциональностью, что позволяет повысить надежность информации, с которой работают специалисты-практики по использованию процессов SAM и поставщики инструментария. Такие дополнительные элементы необходимо использовать в соответствии с требованиями данного раздела.

В теге идентификации программного обеспечения могут присутствовать расширенные элементы (см. 8.5), для того чтобы разрешить включение дополнительных значений, которые ранее не были предопределены. Расширенные элементы должны иметь формат XML и должны включать ссылку XSD, которую можно использовать для проверки информации, содержащейся в этом разделе.

В отличие от обязательного и дополнительного разделов, которых может быть только один, в теге могут присутствовать несколько расширенных разделов. Каждый расширенный раздел должен быть привязан к конкретному создателю или модификатору тега и не должен разрешать смешанное использование. Например, создатель тега может пожелать включить расширенные элементы в собственный инструментарий для обнаружения тегов. Организация потребителя программного обеспечения может пожелать включить внешние элементы, относящиеся общим политикам и процедурам обеспечения жизненного цикла программного обеспечения.

8.2 Имена элементов

Содержание тега идентификации программного обеспечения должно определяться в соответствии с именами элементов, перечисленных в 8.3 и 8.4 (см. ниже). Следование правилам именования позволяет обеспечить согласованную интерпретацию содержания тега идентификации программного обеспечения, независимо от обязательного или дополнительного характера элемента.

Если элемент является дополнительным, его описание все равно должно подчиняться требованиям к именованию и ограничениям, оговоренным в определениях раздела 8.4.

8.3 Обязательные элементы

8.3.1 Индикатор необходимости выверки прав на использование ('entitlement_required_indicator')

XML-тег	entitlement_required_indicator
Тип	булевский
Определение	Данный элемент представляет собой булевский тег, означающий, должны ли права на использование программного обеспечения соответствовать этому элементу, чтобы выверка программного обеспечения могла считаться успешной. Например, для легальной установки и использования программного обеспечения Open Source применение данных о правах на использование программного обеспечения в процессе выверки может не быть «обязательным». Однако это не означает, что с программным обеспечением не связаны права на использование программного обеспечения; это означает лишь то, что для выполнения процесса сверки данные о правах на использование программного обеспечения, оговоренные в системе SAM, не требуются. Таким образом, специалист-практик может управлять каждым программным обеспечением в индивидуальном порядке и фокусироваться только на тех элементах, соответствие которых должно обеспечиваться в официальном порядке. Это не означает, что организация не будет заниматься вопросами соответствия элементов, например, программного обеспечения open source или бесплатно распространяемых продуктов (freeware). Просто решение принимает сама организация. Этот элемент встречается в теге идентификации программного обеспечения только один раз
Пример	<entitlement_required_indicator>true</entitlement_required_indicator>

8.3.2 Наименование продукта ('product_title')

XML-тег	product_title
Тип	Символьная XML-строка
Определение	Название продукта, присвоенное ему создателем программного обеспечения. Это значение, в основном, используется в отчетах для конечных пользователей или отчетах о вычислительных устройствах; использование этого значения в процессе выверки в общем случае не предполагается. Этот элемент встречается в теге идентификации программного обеспечения только один раз
Пример	<product_title>Viewmaster Standard</product_title>

8.3.3 Версия продукта ('product_version')

XML-тег	product_version
Тип	Комплексный тип
Определение	Версия продукта, определяемая как два элемента – числовая версия и имя версии. Данный элемент позволяет создателям программного обеспечения выдавать только числовую информацию о номере версии, которая может использоваться для целей сравнения с информацией о правах на использование программного обеспечения, а также с целью группирования. Кроме того, также имеется строковая версия, чтобы предоставить создателям программного обеспечения возможность указывать любые текстовые фрагменты, которые могут быть предназначены для конечных пользователей, изучающих отчет. Хорошим примером может служить Microsoft Vista®. Большинство специалистов-практиков по использованию процессов SAM немедленно узнают текстовое написание этой версии операционной системы, так как оно представлено в виде Microsoft Vista®, Version SP1. С другой стороны, лишь немногие специалисты-практики по использованию процессов SAM могут распознать, что скрывается за числовой версией 6001.18063. Каждый элемент является независимым, но они должны взаимоотноситься и быть согласованы друг с другом.

Окончание таблицы

<table>
<tr><td>Определение</td><td colspan="3">Номер числовой версии состоит из четырех уровней: основной номер версии, дополнительный номер версии плюс номер сборки и номер для обслуживания. Если поставщик решит не использовать все доступные уровни, значения неиспользуемых уровней должны быть установлены в 0. Предполагается, что числовая версия будет использоваться для целей сравнения с информацией о правах на использование программного обеспечения на этапе выверки процесса управления программными активами.
Строковый вариант версии продукта может содержать как буквенные, так и цифровые символы. Строковый вариант версии продукта является более дружественным, чем ее числовой вариант. Это значение обычно используется в отчетах для конечных пользователей или отчетах по вычислительным устройствам.
Этот элемент встречается в теге идентификации программного обеспечения только один раз</td></tr>
<tr><td rowspan="3">Структура данных</td><td>XML-тег</td><td>Тип</td><td>Определение</td></tr>
<tr><td>имя</td><td>Символьная XML-строка
Одно вхождение</td><td>Текстовое имя версии</td></tr>
<tr><td>численный</td><td>ProductVersionComplexType
Комплексный тип, состоящий из четырех элементов, имеющих числовые значения: «major», «minor», «build», «review»
Одно вхождение</td><td>Числовой идентификатор версии</td></tr>
<tr><td>Пример</td><td colspan="3"><product_version>
<name>10.2 Fix Pack 1</name>
<numeric>
<major>10</major>
<minor>2</minor>
<build>0</build>
<review>0</review>
</numeric>
</product_version>
или
<product_version>
<name>6.2.1279.00</name>
<numeric>
<major>6</major>
<minor>2</minor>
<build>1279</build>
<review>0</review>
</numeric>
</product_version></td></tr>
</table>

8.3.4 Идентификационные данные создателя программного обеспечения ('software_creator')

<table>
<tr><td>XML-тег</td><td colspan="3">software_creator</td></tr>
<tr><td>Тип</td><td colspan="3">Комплексный тип – EntityComplexType</td></tr>
<tr><td>Определение</td><td colspan="3">Этот элемент позволяет процессу обнаружения идентифицировать конкретного создателя программного обеспечения, выпустившего программный пакет.
Имена создателей программного обеспечения в разных странах могут быть точно такими же, однако относиться к разным юридическим лицам. Чтобы обеспечить уникальность, данный элемент должен содержать идентификатор regid создателя программного обеспечения, а также его имя.
Этот элемент встречается в теге идентификации программного обеспечения только один раз</td></tr>
<tr><td rowspan="2">Структура данных</td><td>XML-тег</td><td>Тип</td><td>Определение</td></tr>
<tr><td>имя</td><td>Символьная XML-строка
Одно вхождение</td><td>Этот элемент указывает имя объекта, определенного в теге. Это имя должно быть согласовано между программными продуктами и релизами программного обеспечения</td></tr>
</table>

Окончание таблицы

Структура данных	regid	тип regid Одно вхождение	Идентификатор regid создателя программного обеспечения (см. 6.1.3.) Если объ ект неизвестен или больше не занимается бизнесом, это значение может быть установлено в «unknown» (неизвестно)
Пример	<software_creator> <name>Example Corp</name> <regid>regid.1995-09.com.example</regid> </software_creator>		

8.3.5 Идентификационные данные лицензиара программного обеспечения ('software_licensor')

XML-тег	software_licensor		
Тип	Комплексный тип – EntityComplexType		
Определение	Этот элемент позволяет процессу обнаружения идентифицировать конкретного лицензиара программного обеспечения, владеющего авторскими правами на программный пакет. Имена лицензиаров программного обеспечения в разных странах могут быть точно такими же, однако относиться к разным юридическим лицам. Чтобы обеспечить уникальность, данный элемент должен содержать идентификатор regid лицензиара программного обеспечения, а также его имя. Этот элемент встречается в теге идентификации программного обеспечения только один раз		
Структура данных	XML-тег	Тип	Определение
	имя	Символьная XML-строка Одно вхождение	Этот элемент указывает имя объекта, определенного в теге. Это имя должно быть согласовано между программными продуктами и релизами программного обеспечения
	regid	Символьная XML-строка Одно вхождение	Идентификатор regid лицензиара программного обеспечения (указанного в 6.1.3 Если объект неизвестен или больше не занимается бизнесом, это значение может быть установлено в «unknown» (неизвестно)
Пример	<software_licensor> <name>Example Corp</name> <regid>regid.1995-09.com.example</regid> </software_licensor>		

8.3.6 Уникальный идентификатор программного обеспечения ('software_id')

XML-тег	software_id
Тип	Комплексный тип
Определение	Элемент software_id содержит информацию, которую можно использовать для ссылки на конкретную версию конкретного продукта. Этот элемент требует от создателя тега обеспечить уникальность unique_id для каждого названия и каждой версии программного обеспечения. Различные уровни обновления программного пакета должны различаться уникальными идентификаторами программного обеспечения. Чтобы избежать необходимости привлечения внешнего регистрирующего органа, каждый создатель тега должен использовать собственный идентификатор regid, как это определено в 6.1.3. Идентификатор предоставляется вместе с уникальным ID (unique_id), содержащимся в этом идентификаторе regid. Различные платформы и/или среды разработки могут иметь различные методы создания уникальных ID. Элемент unique_id может быть идентификатором GUID или обычной ссылкой в среде разработки. Например, организация может решить, что ее unique_id может выглядеть, например, так: <productname>_<version>_<releaseID>. Несколько создателей тега могут создавать собственные уникальные software_id для одного и того же программного продукта. Это может происходить, когда создатель программного обеспечения не создал тег идентификации программного обеспечения (например, как это бывает для устаревшего программного обеспечения), и несколько конкурентных сервисных организаций затем создают собственные теги для использования с этим программным обеспечением. Этот элемент встречается в теге идентификации программного обеспечения только один раз

Окончание таблицы

Структура данных	XML-тег	Тип	Определение
	tag_creator_regid	Символьная XML-строка Одно вхождение	Этот элемент определяет идентификационные данные организации, создавшей тег. Идентификатор regid создателя тега (см. в 6.1.3). Обратите внимание, что элементы tag_creator_regid и software_creator_regid могут иметь одни и те же значения – это может быть в случае, когда создатель программного обеспечения создает теги самостоятельно. Включение элемента tag_creator_regid позволяет обеспечить уникальность software_id и также позволяет специалистам-практикам по использованию процессов SAM и инструментарию SAM идентифицировать источник происхождения любого обнаруженного тега идентификации программного обеспечения. Данный элемент не должен содержать символы, которые запрещается использовать в имени файла, например, '/', '\', '[' и пр.
	unique_id	Символьная XML-строка Одно вхождение	Уникальный ID, идентифицирующий конкретную версию конкретного продукта. Элемент unique_id должен удовлетворять ограничениям на использование символов в URI, как это определено в стандарте IETF RFC 3986, раздел 2, Символы. Кроме того, данный элемент не должен содержать символы, которые запрещается использовать в имени файла, например, '/', '\', '[' и пр.
Пример	<software_id> <unique_id>fc3cc419-b5a1-9f16-ed203e537c40</unique_id> <tag_creator_regid>regid.1986-12.com.adobe</tag_creator_regid> </software_id> или <software_id> <unique_id>com.adobe.Acrobat-3D-Win-Multilingual-8.00</unique_id> <tag_creator_regid>regid.1986-12.com.adobe</tag_creator_regid> <software_id>		

8.3.7 Идентификационные данные создателя тега ('tag_creator')

XML-тег	tag_creator		
Тип	Комплексный тип – EntityComplexType		
Определение	Этот элемент позволяет процессу обнаружения идентифицировать конкретного создателя тега для программного пакета. Имена создателей тега в разных странах могут быть точно такими же, однако относиться к разным юридическим лицам. Чтобы обеспечить уникальность, данный элемент должен содержать идентификатор regid создателя тега, а также его имя. Этот элемент встречается в теге идентификации программного обеспечения только один раз		
Структура данных	XML-тег	Тип	Определение
	имя	Символьная XML-строка Одно вхождение	Этот элемент указывает имя объекта, определенного в теге. Это имя должно быть согласовано между программными продуктами и релизами программного обеспечения
	regid	Символьная XML-строка Одно вхождение	Идентификатор regid лицензиара программного обеспечения (указанного в 6.1.3). Если объект неизвестен или больше не занимается бизнесом, это значение может быть установлено в «unknown» (неизвестно)
Пример	<tag_creator> <name>Example Corp</name> <regid>regid.1995-09.com.example</regid> </tag_creator>		

8.4 Дополнительные элементы

8.4.1 Аннотация ('abstract')

XML-тег	abstract		
Тип	Комплексный тип		
Определение	Краткая аннотация, представляющая собой информацию для программного пакета, к которому применяется этот тег, в том числе языковый компонент, обеспечивающий многоязыковую поддержку. Элемент аннотации может встречаться в теге идентификации программного обеспечения несколько раз, однако только один раз для каждого определенного языка. Если язык не указан, в качестве языка устанавливается английский («en»). Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз		
Структура данных	XML-тег	Тип	Определение
	lang	Символьная XML-строка. Представляет собой дополнительный атрибут тега	Язык, на котором составлена аннотация. Языки указываются в соответствии определениями стандарта IETF RFC 4646 (см. gost.e-ecolog.ru)
Пример	<abstract lang=»en»>The View Master software enables viewing of all kinds of document formats.</abstract>		

8.4.2 Принадлежность компонента ('component_of')

XML-тег	component_of		
Тип	Комплексный тип		
Определение	Component_of представляет собой элемент, используемый для отображения отношений дочернего к родительскому объекту между пакетами (т.е. какому родительскому объекту принадлежит этот пакет). В общем случае этот элемент будет использоваться при установке дополнительных компонентов, относящихся к уже существующему на вычислительном устройстве пакету. Этот элемент не используется для определения набора, а используется в том случае, когда устанавливается пакет, добавляющий функциональность к уже существующему на вычислительном устройстве пакету. Для определения группирования продуктов обычно может использоваться либо элемент component_of, либо элемент complex_of, но не оба одновременно. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз		
Структура данных	XML-тег	Тип	Определение
	software_id	Тип идентификации программного обеспечения, описанный в 8.3.5. От одного до неограниченного количества вхождений	Список уникальных идентификаторов программного обеспечения, определяющих отношения принадлежности между данным пакетом и родительским пакетом. Список уникальных идентификаторов программного обеспечения, определяющих отношения принадлежности между данным пакетом и родительским пакетом
Пример	<component_of> <software_id> <unique_id>fc3cc419-b5a1-9f16-ed203e537c40</unique_id> <tag_creator_regid>regid.1986-12.com.adobe</tag_creator_regid> </software_id> </component_of> или <component_of> <software_id> <unique_id>com.adobe.Acrobat-3D-Win-Multilingual-8.00</unique_id> <tag_creator_regid>regid.1986-12.com.adobe</tag_creator_regid> </software_id> </component_of>		

8.4.3 Список компонентов ('complex_of')

<table>
<tr><td>XML-тег</td><td colspan="3">complex_of</td></tr>
<tr><td>Тип</td><td colspan="3">Комплексный тип</td></tr>
<tr><td>Определение</td><td colspan="3">Элемент Complex_of определяет отношения дочерних объектов для данного пакета (т.е. какие пакеты принадлежат данному пакету). Этот элемент обычно используется для выдачи списка продуктов, являющихся частью «набора». Этот элемент представляет собой список уникальных идентификаторов, представляющих продукты, составляющие набор. Для определения группирования продуктов обычно может использоваться либо элемент complex_of, либо элемент component_of, но не оба одновременно.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз.</td></tr>
<tr><td rowspan="2">Структура данных</td><td>XML-тег</td><td>Тип</td><td>Определение</td></tr>
<tr><td>software_id</td><td>Тип идентификации программного обеспечения, описанный в 8.3.5
От одного до неограниченного количества вхождений</td><td>Уникальный идентификатор программного обеспечения, определяющий отношения принадлежности между данным пакетом и его дочерним пакетом. Данный элемент обычно используется при определении пакетов, составляющих набор.</td></tr>
<tr><td>Пример</td><td colspan="3"><complex_of>
<software_id>
<unique_id>fc3cc419-b5a1-9f16-ed203e537c40</unique_id>
<tag_creator_regid>regid.1986-12.com.adobe</tag_creator_regid>
</software_id>
<software_id>
<unique_id>a584c19-b5a1-9f16-ed203e5ab45fc</unique_id>
<tag_creator_regid>regid.1986-12.com.adobe</tag_creator_regid>
</software_id>
</complex_of>
или
<complex_of>
<software_id>
<unique_id>com.adobe.Acrobat-3D-Win-Multilingual-8.00</unique_id>
<tag_creator_regid>regid.1986-12.com.adobe</tag_creator_regid>
</software_id>
</complex_of></td></tr>
</table>

8.4.4 Источник данных ('data_source')

<table>
<tr><td>XML-тег</td><td>data_source</td></tr>
<tr><td>Тип</td><td>Символьная XML-строка</td></tr>
<tr><td>Определение</td><td>Основа источника данных для окончательной установки.
Значениями данного элемента могут быть, в частности, следующие строки (но не ограничиваясь ими): CD, MSDN CD, Библиотека электронного распространения и Библиотека эталонного программного обеспечения – Выпущено для распространения. Указывая эти значения в процессе работы с тегами идентификации программного обеспечения, специалисты-практики по использованию процессов SAM могут быстро оценить, какие элементы конфигурации программного обеспечения установлены на каких платформах, и каким образом была совершена установка каждого такого элемента. Данный элемент не требует проведения нормализации между различными создателями тегов, поскольку многие организации имеют собственные определения типов источников данных, и для специалиста-практика по использованию процессов SAM данная информация носит лишь уведомительный, но не обязательный характер.
Значения, используемые в данном элементе, должны быть согласованы внутри организации и по линейкам продуктов.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз</td></tr>
<tr><td>Пример</td><td><data_source>DVD</data_source></td></tr>
</table>

8.4.5 Зависимость ('dependency')

XML-тег	dependency		
Тип	Комплексный тип		
Определение	Данный элемент предоставляется для того, чтобы разрешить программному обеспечению указывать, что для запуска ему необходим другой продукт. Такая необходимость никак не связана с лицензированием программного обеспечения или правами на использование программного обеспечения; это просто требование для запуска программного обеспечения. Например, приложение Java™ может нормально запускаться только в определенной версии Java™. Шаблон Excel® требует Excel®. Эти зависимости не обязательно являются строгими зависимостями, используемыми программным обеспечением для проверки наличия конкретного тега идентификации программного обеспечения до того, как будет запущено приложение, а, скорее, являются инструкциями, выдаваемыми в базу данных SAM для получения сведений об отношениях программного обеспечения и/или, возможно, для предоставления информации в среду службы поддержки. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз.		
Структура данных	XML-тег	Тип	Определение
	software_id	Тип идентификации программного обеспечения, описанный в 8.3.5 От одного до неограниченного количества вхождений	Уникальный идентификатор программного обеспечения, определяющий тег зависимости
Пример	<dependency> <software_id> <unique_id>fc3cc419-b5a1-9f16-ed203e537c40</unique_id> <tag_creator_regid>regid.1986-12.com.adobe</tag_creator_regid> </software_id> </dependency>		

8.4.6 Список владельцев элемента ('elements_owner')

XML-тег	elements_owner		
Тип	Комплексный тип		
Определение	Этот элемент обеспечивает возможность указывать, кто заявляет о владении элементами в теге. Такое заявление о владении не является официальным (как, например, цифровая подпись), тем не менее, оно содержит ценную информацию для модификаторов тегов. Если оговорено, что элемент находится «во владении», это означает что значение, указанное в элементе, не должно меняться, пока такое изменение не будет согласовано с существующим владельцем элемента. Создатель тега должен указать элементы, которые не должны изменяться никакими модификаторами тегов. В элементе должны использоваться ID элемента, поддерживаемые XSD. Эти ID элемента создаются создателем тега и должны быть уникальны для каждого тега идентификации программного обеспечения . ID используются для ссылок на другие элементы тега внутри самого тега. Способы использования ID приведены в примере ниже, а также в 6.1.12. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз		
Структура данных	XML-тег	Тип	Определение
	owner_regid	Символьная XML-строка Одно вхождение	Если определен элемент elements_owner, эта составляющая определяет компанию, являющуюся владельцем значений. В общем случае в коммерческом, т. е. готовом к использованию программном обеспечении (COTS), в качестве полного определения владельца используется идентификатор regid. Элемент owner_name также может указываться в приложениях, создаваемых на внутреннем уровне организации

Окончание таблицы

Структура данных	owner_name	Символьная XML-строка Ни одного или одно вхождение	Данный элемент предоставляет более подробную информацию о том, кто является владельцем значений, указанных в теге идентификации программного обеспечения. В общем случае элемент owner_name используется только в приложениях, создаваемых на внутреннем уровне организации, и указывается для того, чтобы можно было связаться с физическим лицом или группой лиц. При желании в коммерческих приложениях элемент owner_name также может использоваться для указания дополнительных контактных данных, относящихся к элементам, находящимся во владении.
	element_id	Символьная XML-строка От ни одного до неограниченного количества вхождений	В элементе element_id приводится список ID, владельцем которых является конкретный владелец. ID указываются создателем элемента и должны быть уникальны для конкретного тега, однако требование обеспечения уникальности между различными тегами не устанавливается, поскольку эти ID используются только для отображения связей отношений внутри тега или для ссылки на конкретные элементы внутри указанного тега. Ниже приводится пример оформления и использования ID в структуре elements_owner
Пример	<swid:entitlement_required_indicator ID="e8_3_1">true</swid:entitlement_required_indicator> <swid:product_title ID="e8_3_2">Adobe Photoshop CS3</swid:product_title> <swid:product_version ID="e8_3_3"> <swid:name ID="e8_3_3sub1">10.2</swid:name> <swid:numeric ID="87_3_3sub2"> <swid:major ID="e8_3_3sub2sub1">10</swid:major> <swid:minor ID="e8_3_3sub2sub2">2</swid:minor> <swid:build ID="e8_3_3sub2sub3">0</swid:build> <swid:review ID="e8_3_3sub2sub4">0</swid:review> </swid:numeric> </swid:product_version> <swid:software_creator ID="e8_3_4"> <swid:name>Adobe Systems Incorporated</swid:name> <swid:regid>regid.1986-12.com.adobe</swid:regid> </swid:software_creator> <swid:software_licensor ID="e8_3_5"> <swid:name>Adobe Systems Incorporated</swid:name> <swid:regid>regid.1986-12.com.adobe</swid:regid> </swid:software_licensor> <swid:software_id ID="e8_3_6"> <swid:unique_id>Photoshop-CS3-Win-GM-en_US</swid:unique_id> <swid:tag_creator_regid>regid.1986-12.com.adobe</swid:tag_creator_regid> </swid:software_id> <swid:tag_creator ID="e8_3_7"> <swid:name>Adobe Systems Incorporated</swid:name> <swid:regid>regid.1986-12.com.adobe</swid:regid> </swid:tag_creator> <!-- Optional elements --> <swid:elements_owner> <swid:owner_name>Adobe Systems Inc. </swid:owner_name> <swid:owner_regid>regid.1986-12.com.adobe</swid:owner_regid> <swid:elements_ID>e8_3_1</swid:elements_ID> <swid:elements_ID>e8_3_2</swid:elements_ID> <swid:elements_ID>e8_3_3</swid:elements_ID> <swid:elements_ID>e8_3_4</swid:elements_ID> <swid:elements_ID>e8_3_5</swid:elements_ID> <swid:elements_ID>e8_3_6</swid:elements_ID> <swid:elements_ID>e8_3_7</swid:elements_ID> </swid:elements_owner>		

8.4.7 Сведения об установке ('installation_details')

<table>
<tr><td>XML-тег</td><td colspan="3">installation_details</td></tr>
<tr><td>Тип</td><td colspan="3">Комплексный тип</td></tr>
<tr><td>Определение</td><td colspan="3">В этом элементе указывается конкретная информация о полном пути к каталогам размещения тегов идентификации программного обеспечения для конкретной установки программного пакета, а также сведения об экземпляре установки. При каждой установке программного обеспечения в систему будет добавлено два тега идентификации программного обеспечения – один в общий системный каталог платформы (как это указано в 6.1.4), а другой – в корневой каталог установленного программного пакета.
Настоятельно рекомендуется, чтобы в данный элемент в любом случае вносились сведения о каталогах размещения тегов, так как это позволит поставщикам инструментария SAM сопоставлять оба тега идентификации программного обеспечения как связанные между собой.
На платформах, обеспечивающих легкий перенос программных продуктов из одного места в другое (например, на платформе Apple Macintosh™), настоятельно рекомендуется, чтобы программное приложение регулярно проверяло правильность определения элемента installation_details, и чтобы поставщики инструментария SAM проверяли правильность каталогов, в которых происходит обнаружение тегов, и сравнивали эти данные с данными, содержащимися в элементе installation_details.
Экземпляры установки указываются для программного обеспечения, которое может устанавливаться на одной платформе несколько раз. Несколько экземпляров могут устанавливаться в тех случаях, когда установки осуществляются для конкретных конечных пользователей, или когда для работы конкретного конечного пользователя или, в общем случае, системы, требуются несколько копий конкретного программного пакета.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз</td></tr>
<tr><td>Структура данных</td><td>XML-тег</td><td>Тип</td><td>Определение</td></tr>
<tr><td></td><td>location_platform</td><td>Символьная XML-строка
Ни одного или одно вхождение</td><td>Этот элемент представляет собой полный путь к каталогу, в котором размещается общий тег идентификации программного обеспечения. Если общий путь находится в указанном системном каталоге (например, в случае Microsoft Vista®), значением данного элемента может быть строка «system»</td></tr>
<tr><td></td><td>location_installation</td><td>Символьная XML-строка
Ни одного или одно вхождение</td><td>Этот элемент представляет собой полный путь к тегу идентификации программного обеспечения, установленному в корневой каталог программного пакета</td></tr>
<tr><td></td><td>installation_instance</td><td>Символьная XML-строка
Ни одного или одно вхождение</td><td>Если программное обеспечение допускает несколько установок, элемент installation_instance позволяет организациям выдавать уникальный идентификатор для каждой установки.
Например, если для отдельных конечных пользователей могут создаваться несколько экземпляров установки, каждая установка может быть идентифицирована id конечного пользователя. Если программное обеспечение можно устанавливать несколько раз для одного и того же пользователя или системы, таким идентификатором может быть обычное число, увеличивающееся на единицу по мере обнаружения других тегов идентификации программного обеспечения в процессе установки.
Данный элемент не должен содержать символы, которые запрещается использовать в имени файла, например, '/', '\', '[' и пр.</td></tr>
<tr><td></td><td>installation_locale</td><td>Символьная XML-строка
От ни одного до неограниченного количества вхождений</td><td>Этот элемент определяет языковый стандарт или стандарты, поддерживаемые установленным программным обеспечением. Языковые стандарты определяются в документе IETF RFC 4646 (см. gost.e-ecolog.ru /rfc/rfc4646.txt). Если установленная версия программного обеспечения поддерживает несколько языковых стандартов, это может быть отражено в теге идентификации программного обеспечения, который будет содержать несколько элементов installation_locale</td></tr>
</table>

Окончание таблицы

Структура данных	installation_target_id	Символьная XML-строка Ни одного или одно вхождение	Значение, позволяющее выполнять идентификацию машины, устройства хранения и/или виртуальную среду, на которых было установлено программное обеспечение. Данный элемент не должен содержать символы, которые запрещается использовать в имени файла, например, '/', '\', '[' и пр. Значение, устанавливаемое для данного элемента, определяется на основании относящихся к конкретной установке параметров, предоставляемых программой-установщиком (или самоустанавливающимися программами). В параметре может быть указано либо конкретное значение, либо ссылка на спецификации, которые можно получить посредством выполнения соответствующего запроса к операционной системе. Если издатель программного обеспечения не предоставил спецификации, используется значение по умолчанию. В качестве значения по умолчанию рекомендуется устанавливать серийный номер носителя (на котором было установлено программное обеспечение). Элемент installation_target_id может содержать номер актива, серийный номер носителя, на котором установлено программное обеспечение, MAC-адрес сетевой карты или другую уникальную ссылочную информацию. Необходимо понимать, что в некоторые из этих значений могут вноситься изменения (с сохранением фактических идентификационных данных), например, может быть заменена сетевая карта, что приведет к изменению ее MAC-адреса. Процедуры управления программными активами должны распознавать такие случаи. Данное значение является рекомендуемым компонентом установленной версии имени файла тега идентификации программного обеспечения. (см. 6.1.7) Предполагается, что правила, определяющие типы значений, указываемых в этом элементе, будут со временем меняться, и что эти правила будут поддерживаться провайдерами платформ
Пример	<installation_details> <location_platform>C:\Documents and Settings\All Users\Application Data \adobe\adobe.com-photoshop8.0pro.swidtag</location_platform> <location_installation>C:\Program Files\Adobe\Photoshop CS\adobe.com-photoshop8.0pro.swidtag</location_installation> <installation_instance>1<\installation_instance> <installation_target_id>0018F8096CE1</installation_target_id> <installation_locale>en-US</installation_locale> <installation_locale>en-GB</installation_locale> <installation_locale>en-AU</installation_locale> </installation_details>		

8.4.8 Ключевые слова ('keywords')

XML-тег	Ключевые слова
Тип	Комплексный тип
Определение	Этот элемент позволяет создателю или модификатору тега добавлять в тег идентификации программного обеспечения конкретные ключевые слова. Значения ключевых слов в данной части стандарта ISO/IEC 19770 не определяются. Значения ключевых слов приводятся в настоящем документе как способ, с помощью которого со здатели или модификаторы тега облегчают работу поисковых механизмов при обнаружении ими тегов идентификации программного обеспечения, относящихся к конкретному объекту. Модификаторы тегов могут добавлять отдельные ключевые слова; ключевые слова могут указываться в элементе «elements_owner»; ключевые слова также могут быть подписаны. Поскольку элемент ключевых слов допускает наличие нескольких подэлементов ключевых слов, каждый такой подэлемент может находиться во владении или быть подписан своим индивидуальным владельцем

Окончание таблицы

Структура данных	XML-тег	Тип	Определение
	Keyword	Символьная XML-строка От ни одного до неограниченного количества вхождений	Этот элемент используется для указания ключевых слов, применяемых к конкретному тегу идентификации программного обеспечения. Ключевые слова заносятся по одному, могут идентифицироваться как находящиеся во владении (владельцев, определенных в элементе elements_owner, 8.4.6) и также могут быть включены в цифровую подпись (см. Аутентичность тегов идентификации программного обеспечения, 6.1.13). Список ключевых слов могут пополнять несколько модификаторов тегов (каждый добавляет собственные ключевые слова)
Пример	<keywords> <keyword>Acme</keyword> <keyword>Painter</keyword> </keywords>		

8.4.9 Информация о лицензии и канале ('license_linkage')

XML-тег	license_linkage		
Тип	Комплексный тип		
Определение	В этом элементе представлена информация, которую можно использовать для определения надлежащей структуры прав на использование программного обеспечения для установки продукта, относящегося к данному тегу. Элементы, являющиеся частью элемента license_linkage, содержат информацию о том, как может быть установлен продукт, информацию о текущем состоянии лицензии на этот продукт для конкретной системы, на которой обнаружен тег. Примечание – Состояние лицензии не связано напрямую с правами на использование программного обеспечения. Данные о состоянии лицензии используются для установки конкретного программного пакета на конкретную машину. Права на использование, с другой стороны, определяют юридическое право собственности в отношении прав на использование программного обеспечения по лицензии. Сведение о правах на использование приведены в части 3 стандарта ISO/IEC 19770. Элементы, содержащиеся в элементе license_linkage, помогают специалистам-практикам по использованию процессов SAM быстро идентифицировать установку несанкционированного программного обеспечения в своем окружении. Наличие этих тегов не является обязательным требованием, тем не менее, они могут оказать помощь специалистам-практикам по использованию процессов SAM, предоставляя им более подробную информацию о том, из какого источника получено программное обеспечение. Определяя данные теги, специалисты-практики по использованию процессов SAM могут создавать правила, с помощью которых они смогут управлять каждым изменением по отдельности, а не отслеживать все изменения, которые могут происходить в их рабочем окружении. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз.		
Структура данных	XML-тег	Тип	Определение
	activation_status	Символьная XML-строка От ни одного до неограниченного количества вхождений	Значения, содержащиеся в данном элементе, относятся к различным уровням лицензирования, которые может отслеживать конкретный лицензиар программного обеспечения для отдельной машины. Каждый лицензиар программного обеспечения может иметь свой собственный набор значений состояния, но, по мере возможности, для каждого лицензиара программного обеспечения эти значения должны быть единообразны. Представления этих значений могут быть следующими: a) trial (пробное использование) – означает, что программное обеспечение работает в режиме пробного использования. Данное значение может включать в себя количество дней, которое действует режим пробного использования, или указание на то, что период пробного использования завершен; b) serialized (указан серийный номер) – означает, что конечный пользователь или потребитель программного обеспечения в процессе установки указали действительный серийный номер, однако продукт еще не был активирован ими;

Продолжение таблицы

Структура данных			c) fully licensed (полностью лицензирован) – означает, что продукт был активирован и, с точки зрения лицензиара программного обеспечения, данная установка является полностью функциональной установкой для конкретной системы; d) unlicensed (не лицензирован) – означает, что программное обеспечение более не может запускаться на устройстве или может запускаться, но в режиме с ограничениями. Программное обеспечение может перейти в это состояние в следующих случаях: 1) завершен период пробного использования, 2) завершен срок действия лицензии, выдаваемой на определенное время, 3) для пакета был указан серийный номер, но пакет не был активирован за определенный промежуток времени. Значения, используемые в данном элементе, должны быть согласованы внутри организации и по линейкам продуктов
	channel_type	Символьная XML-строка От ни одного до неограниченного количества вхождений	В этом элементе представлена информация о том, для какого канала было предназначено это конкретное программное обеспечение. Значения, используемые в данном элементе, могут быть уникальными для производителя программного обеспечения, однако должны быть согласованы с аналогичными значениями других продуктов, публикуемых конкретным поставщиком. Представления этих значений могут быть следующими: a) volume (оптовый); b) retail (розничный); c) OEM; d) academic (академический). Такие значения, предоставленные лицензиаром программного обеспечения, могут использоваться специалистами-практиками по использованию процессов SAM для идентификации программного обеспечения, которое может быть установлено в операционном окружении организации, однако не соответствует организационной политике. Например, программное обеспечение, которое было предназначено для академического канала, в общем случае, не подходит для установки в корпоративном окружении. Значения, используемые в данном элементе, должны быть согласованы внутри организации и по линейкам продуктов
	channel_name	Символьная XML-строка Ни одного или одно вхождение	В этом элементе указывается каталог размещения имени канала. Это позволяет организациям реселлеров создавать теги идентификации программного обеспечения, включающие имя партнера по продажам или распространению
	customer_type	Символьная XML-строка От ни одного до неограниченного количества вхождений	Тип пользователя определяет целевого пользователя, но не канал. Значения, используемые в данном элементе, могут быть уникальными для производителя программного обеспечения, однако должны быть согласованы с аналогичными значениями других продуктов, публикуемых конкретным поставщиком. Представления этих значений могут быть следующими: a) government (государственный); b) corporate (корпоративный); c) educational (образовательный); d) retail (розничный). Учитывая текущие права на использование программного обеспечения, продукты, ориентированные на различных пользователей, часто могут иметь различные стоимости. Кроме того, на процессы установки таких продуктов могут накладываться ограничения, например экземпляр программного обеспечения, созданного для пользователя, относящегося к категории educational (образовательный), обычно реализуется по более низкой стоимости и часто не лицензируется для использования в корпоративном окружении. Значения, используемые в данном элементе, должны быть согласованы внутри организации и по линейкам продуктов

Окончание таблицы

Пример	<license_linkage> <activation_status>Fully Licensed</activation_status> <channel_type>Volume</channel_type> <channel_name>Reseller name</channel_name> <customer_type>Corporate</customer_type> </license_linkage> или <license_linkage> <activation_status>Trial</activation_status> <channel_type>Retail</channel_type> <customer_type>Retail</customer_type> </license_linkage>

8.4.10 «Отпечаток» пакета ('package_footprint')

XML-тег	package_footprint		
Тип	Комплексный тип		
Определение	Определяет набор файлов и других записей, означающих, что продукт установлен. Также содержит ссылку на элемент package_footprint из внешнего URI. На платформе Windows® к «другим записям» могут относиться записи реестра, записи WMI, а также данные MSI. На других платформах в элемент может включаться дополнительная информация, относящаяся к платформе (добавляется по желанию). Информация, содержащаяся в элементе package_footprint, может использоваться инструментарием SAM и специалистами-практиками по использованию процессов SAM для обеспечения степеней достоверности информации, представленной в теге идентификации программного обеспечения и перекрёстных ссылок. Упомянутая информация и ссылки должны быть указаны создателем программного обеспечения и представлены в сравнении с фактически обнаруженными в процессе инвентаризации. Обратите внимание, что элемент package_footprint не предполагается использовать ни для проверки завершения установки программного обеспечения, ни для проверки фактического запуска этого программного обеспечения на выполнение. Для проверки того, что тег идентификации программного обеспечения установлен на устройство, на котором уже фактически установлено программное обеспечение, могут использоваться составляющие, перечисленные в разделе основных составляющих. Дополнительные и прочие составляющие предоставляются создателем тега с целью облегчения работы инструментария SAM и специалистов-практиков по использованию процессов SAM. Используя информацию «отпечатка» для приложения, инструментарий SAM и специалисты-практики по использованию процессов могут использовать эту информацию для фильтрации обширного списка обнаруженных данных, которые они получают от всех устройств, и могут перейти к практике использования процессов SAM в применении к индивидуальным объектам, так как в их распоряжении будет более точный список объектов, которые разрешены к установке, который можно сравнить со списком того, что обнаружено. Примечание – Примеры различных элементов приведены в 8.6.6 FootprintModuleComplexType, в котором приведены сведения о совместном функционировании различных типов, а также данные о структуре типа, используемой в элементе package_footprint. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз		
Структура данных	XML-тег	Тип	Определение
	external_description	Строка URI Ни одного или одно вхождение	Этот элемент позволяет создателю тега вести список связанных с конкретной системой объектов, доступных для обращения посредством косвенной ссылки на их сайт. Таким образом, создатель тега становится владельцем списка и может вносить необходимые изменения в список, не связывая себя необходимостью распространять патчи или новые версии продукта с новым набором файлов.
	primary	FootprintModule Комплексный тип Ни одного или одно вхождение	Определяет файлы и другие составляющие, считающиеся «основными» для программного пакета. Если на устройстве в основном элементе будут обнаружены составляющие, будет считаться, что имеется высокая вероятность установки программного обеспечения и тег должен считаться действительным в этом программном обеспечении.

Продолжение таблицы

Структура данных			Примечания 1 Имя файла само по себе не является уникальным. Для обеспечения уникальности в этом элементе предусмотрено определение нескольких характеристик, в том числе таких, как имя, размер, md5, версия и «другие» типы, которые может определить создатель тега. 2 В конкретном файле в основном элементе могут иметься несколько записей. Это позволяет использовать один «отпечаток» в нескольких релизах патчей, в которых файлы могут измененные данные размера, версии или контрольных сумм MD5. В этих случаях инструментарию для обнаружения тегов нужно будет обнаружить одно уникальное определение файла для каждого представленного уникального имени файла (т. е. если файл «abc.com» имеет три разных размера и три разных записи MD5, и инструментарий для обнаружения тегов SAM обнаружит соответствие одного обнаруженного файла с одной из этих трех записей, то файл «abc.com» определяется как существующий на устройстве). 3 Если имеется одна версия каждой основной составляющей (файла, а также элемента os_configuration_record или другого элемента типа FootPrintModuleComplex), определенного в основном элементе, то инструментарий SAM и специалисты-практики по использованию процессов SAM должны иметь высокую степень уверенности в том, что тег идентификации программного обеспечения правильно идентифицирует установленное программное обеспечение. Если некоторые основные составляющие не будут обнаружены на устройстве, это будет означать, что возникла исключительная ситуация, и специалист-практик по использованию процессов SAM или администратор релизов долж-ны выяснить, почему на устройстве имеются не все основные составляющие
	secondary	FootprintModule Комплексный тип Ни одного или одно вхождение	Определяет файлы и другие составляющие, считающиеся «вторичными» для программного пакета. Эти составляющие используются не для проверки того, что тег относится к фактически установленному программному обеспечению, а в качестве фильтра алгоритмами распознавания программного обеспечения, определяющими на основе обнаруженных файлов, установлен ли программный пакет. Имея в распоряжении список файлов, которые можно легко «отфильтровать», механизм распознавания программного обеспечения выдает набор файлов с меньшим количеством несовпадений, с которым можно дальше работать. Как и для описанного выше основного элемента, файлы могут содержать специальные характеристики, обеспечивающие уникальность для создателя тега
	related	FootprintModule Комплексный тип Ни одного или одно вхождение	Определяет файлы и другие составляющие, «слабосвязанные» с программным пакетом, которые также могут устанавливаться с конкретным программным пакетом. Эти файловые записи также используются для составления списка файлов, собираемых агентом обнаружения с целью исключения файлов, связанных с «известным» программным обеспечением. Примером может служить программа-установщик, устанавливающая OEM-версию программного пакета. Файлы, присутствующие в установке OEM, должны иметь возможность установления слабой связи с основным пакетом (чтобы их можно было соответствующим образом отфильтровать процессом распознавания программного обеспечения), однако, если на владение этими файлами претендует другой программный пакет, то программное обеспечение, заявляющее о владении, будет иметь приоритет над любым «слабосвязанным» программным обеспечением. К другим слабосвязанным компонентам могут быть отнесены компоненты,

Окончание таблицы

Структура данных			совместно используемые программными пакетами, и/или компоненты, которые могут устанавливаться в качестве дополнительных надстроек к программному пакету
Пример	<package_footprint> <external_description>http://www.adobe.com/acrobat/(sofware_id)/filelist.xml</external_descriptio n> </package_footprint> или <package_footprint> <primary> <file> <name>acrobat.exe</name> <size>349808</size> <version>8.1.0.137</version> <md5>9bb7d80f752d9aba168f7795a5ffa7f6</md5> </file> <os_configuration_record> <record_type>WMI</record_type> <path>Root\CIMV2</path> <name>Win32_Product</name> <internal_path>Name= 'Adobe Acrobat 8 Professional'</internal_path> <entry> <name>Version</name> <value>8.1.2</value> <type></type> </entry> <entry> <name>Vendor</name> <value>Adobe Systems</value> <type></type> </entry> </os_configuration_record> </primary> </package_footprint> Примечание – В файле filelist.xml должна использоваться структура, аналогичная структуре тега идентификации программного обеспечения, а для выдачи определений файлов для конкретного элемента software_id должен использоваться элемент «package_footprint»		

8.4.11 Упаковщик ('packager')

XML-тег	packager		
Тип	Комплексный тип		
Определение	Этот элемент содержит сведения о том, кто именно изменил программный пакет для конкретного набора процедур установки. Этот элемент чаще всего определяется администратором релизов организации, так как программное обеспечение настраивается для установки в этой компании. В этих случаях элемент packager часто ассоциируется с данными таких элементов, как release_id и release_package. Элемент также может использоваться третьими сторонами в тех случаях, когда продукт проходит процедуру OEM и переупаковывается, или если программный пакет настраивается для установки в конкретной конфигурации. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз		
Структура данных	XML-тег	Тип	Определение
	by	Символьная XML-строка Одно вхождение	Атрибут, содержащий информацию о третьей стороне, осуществляющей упаковку продукта. Ниже приводятся примеры основных типов значений: a) название упаковочной компании; b) используемая технология упаковки; c) внутреннее групповое имя упаковщика;

Окончание таблицы

Структура данных			d) используемый инструмент управления настольными системами. Более подробная информация об этой составляющей приведена на веб-сайте: gost.e-ecolog.ru
	part	Символьная XML-строка Одно вхождение	Дополнительная справочная информация о стороннем продукте, например, шифр компонента
Пример	<packager> <by>ACME Widget Corp</by> <part>Photoshop CS3 – OEM'd into widget designer – P#345ABD</part> </packager>		

8.4.12 Категория продукта ('product_category')

XML-тег	product_category		
Тип	Комплексный тип		
Определение	Средство, с помощью которого высокоуровневое подразделение осуществляет классификацию названий продуктов. Стандартизированный перечень категорий/групп приводится в системе стандартов кодирования товаров и услуг Организации Объединенных Наций: UNSPSC (более подробная информация приведена по адресу: http://www.unspsc.org/), номер в товарной номенклатуре 43230000). Коды UNSPSC, приведенные в разделе под номером 43230000 спецификации, применяются в тех случаях, когда обнаруживается большое количество общеупотребительных категорий программного обеспечения. Категоризация продуктов может выполняться с помощью кодов UNSPSC. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз		
Структура данных	XML-тег	Тип	Определение
	UNSPSC_ver	Символьная XML-строка Одно вхождение	Номер версии используемого набора кодов UNSPSC. Обычно для использования кода версия не требуется. Однако, если для предоставления дополнительной функциональности инструментарию необходимо знать версию (например, для предоставления различных имен на одном из 9 других языков), эта версия должна быть предоставлена инструментарию. Примером формата версий UNSPSC может быть 10.0501
	segment_title	Символьная XML-строка Одно вхождение	Имя сегмента, которому принадлежит продукт
	family_title	Символьная XML-строка Одно вхождение	Имя, обеспечивающее распознавание семейства продуктов
	class_title	Символьная XML-строка Одно вхождение	Имя класса
	commodity_title	Символьная XML-строка Одно вхождение	Название продукта
	code	Числовое значение, выраженное 8 цифрами Одно вхождение	Коды должны указываться в соответствии с их определением в списке кодов UNSPSC

Окончание таблицы

Пример	<category> <UNSPSC_ver>10.0501</UNSPSC_ver> <segment_title>Information Technology Broadcasting and Telecommunications</segment_title> <family_title>software</family_title> <class_title>Finance accounting and enterprise resource planning ERP software</class_title> <commodity_title>Enterprise resource planning ERP software</commodity_title> <code>43231602</code> </category>

8.4.13 Семейство продуктов ('product_family')

XML-тег	product_family
Тип	Символьная XML-строка
Определение	Семейство продуктов определяет элемент, который могут использовать издатели и лицензиары программного обеспечения для группирования программных продуктов совместно с отчетами специалистов-практиков по использованию процессов SAM. Примером типа продукта, который может использовать данный элемент, может служить инструментарий резервного копирования, в котором сервисы резервного копирования, резервного копирования сервера и резервного копирования клиентской машины, с которыми работает инструментарий, продаются отдельно как независимые продукты. В этом случае, если все продукты имеют один и тот же определенный элемент product_family, инструментарий SAM может автоматически соответствующим образом группировать данные обнаруженных тегов идентификации программного обеспечения, как это показано ниже: Пример утилиты резервного копирования Сервер резервного копирования – обнаружено 20 установок Утилита резервного копирования сервера – обнаружено 240 установок Утилита резервного копирования клиента – обнаружено 10240 установок
Пример	<product_family>Example Backup Utility</product_family>

8.4.14 Идентификатор продукта ('product_id')

XML-тег	product_id
Тип	Символьная XML-строка
Определение	Идентификация продукта. Не зависит от версии продукта. Элемент product_id должне представлять собой уникальную ссылку, но только в пределах организации производителя программного обеспечения, т. е. этот идентификатор может не быть глобальным уникальным идентификатором. Рекомендуется, чтобы ID продукта отличался от наименования продукта или от других его рыночных названий, поскольку от релиза к релизу они могут меняться. Элемент product_id должен быть идентификатором, который сопровождал бы продукты на протяжении всего их жизненного цикла и не зависел бы от рыночных изменений. Элемент product_id используется для определения линии наследования между продуктами с целью идентификации разрешенных обновлений. Это значение может использоваться или не использоваться при определении прав на использование программного обеспечения. Если в правах на использование программного обеспечения указывается, что продукт может обновляться в течение определенного периода времени, в документе о правах на использование программного обеспечения, естественно, будут отсутствовать данные о том, какие могут применяться и какие будут доступны в будущем имена или версии продукта в течение этого времени. Элемент product_id позволяет указывать в документе о правах на использование программного обеспечения, что сначала разрешается установка конкретной версии продукта, а затем разрешается установка любых обновленных продуктов, имеющих тот же product_id, если значение элемента release_date попадает в диапазон, оговоренный в правах на использование программного обеспечения. Примечание – Записей product_id может быть несколько. Такое может произойти, когда создатель программного обеспечения выпускает новый продукт и разрешает конечным пользователям или потребителям программного обеспечения, пользующимся более старыми продуктами, обновиться до нового продукта. Например: Продукт А product_id = 1234XYZ Продукт В product_id = ABCDPDQ

Окончание таблицы

Определение	Продукт С (этот продукт разрешает производить технические обновления продукта А или продукта В) product_id = 9876HJK <- это новый ID продукта для продукта С... product_id = 1234XYZ product_id = ABCDPDQ Если более поздние релизы Продукта С захотят разрешить технические обновления с более ранних версий Продукта С, в элемент product_id надо будет только включить новый ID для этого продукта – 9876HJK. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз
Пример	<product_id>fc3cc419-b5a1-9f16-ed203e537c40</product_id>

8.4.15 Дата релиза ('release_date')

XML-тег	release_date
Тип	XML-тип dateTime
Определение	Этот тег обычно используется организацией потребителя программного обеспечения в рамках процесса релизов ITIL. Для этой установки был осуществлен релиз элемента конфигурации программного обеспечения Date (Дата). Для упрощения процессов выверки в элементе конфигурации программного обеспечения должна использоваться единая дата релиза. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз
Пример	<release_date>2008-01-21T12:00:00</release_date>

8.4.16 Идентификатор релиза ('release_id')

XML-тег	release_id
Тип	Символьная XML-строка
Определение	Этот тег обычно используется организацией потребителя программного обеспечения в рамках процесса релизов ITIL. Данные, используемые в процессе выверки для идентификации атрибутов пакета релизов при установке и связанных прав на использование программного обеспечения. Записи данного элемента должны быть согласованы по всем тегам идентификации программного обеспечения для любого заданного элемента конфигурации программного обеспечения. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз
Пример	<release_id>COE-Base-Ver 8, 2008-01-21</release_id>

8.4.17 Пакет релизов ('release_package')

XML-тег	release_package		
Тип	Комплексный тип		
Определение	Этот тег обычно используется организацией потребителя программного обеспечения в рамках процесса релизов ITIL. Подтверждающая информация о том, что созданный пакет релизов соответствует системной архитектуре провайдера услуг, спецификациям управления услугами и инфраструктурным спецификациям. Примечание – Программные пакеты конечных пользователей практически всегда настраиваются под требования провайдера услуг, т. е. имеют конкретные параметры установки и/или сочетания программных наборов. (Теги идентификации программного обеспечения релизов полностью независимы от любых тегов идентификации программного обеспечения внешних поставщиков программного обеспечения). Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз		
Структура данных	XML-тег	Тип	Определение
	Sign_off	Символьная XML-строка Одно вхождение	В этой записи указывается лицо, утвердившее правильность упаковки программного обеспечения и готовность его перехода на этап тестирования

Окончание таблицы

Структура данных	Sign_off_ date	XML-тип dateTime Одно вхождение	В этой записи указывается дата утверждения программного пакета
	By	Символьная XML-строка Одно вхождение	В этой записи указывается разработчик программного обеспечения, создавший пакет. Данная информация может использоваться при возникновении вопросов на этапе тестирования
Пример	<release_package> <sign_off>Jane Doe</sign_off> <sign_off_date>2008-01-10T12:00:00</sign_off_date> <by>John Doe</by> </release_package>		

8.4.18 Развертывание релиза ('release_rollout')

XML-тег	release_rollout		
Тип	Комплексный тип		
Определение	Подтверждающая информация, относящаяся к лицу, утвердившему пакет релизов как готовый к использованию в коммерческих целях. Здесь также указывается дата утверждения. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз		
Структура данных	XML-тег	Тип	Определение
	Sign_off	Символьная XML-строка Одно вхождение	В этой записи указывается лицо, утвердившее правильность тестирования программного обеспечения в промышленном окружении и готовность его перехода на этап использования в коммерческих целях
	Sign_off_ date	XML-тип dateTime Одно вхождение	В этой записи указывается дата утверждения опытного образца программного обеспечения
	By	Символьная XML-строка Одно вхождение	В этой записи указывается специалист-практик по использованию процессов SAM, который управлял этапом тестирования продукта в опытном окружении. Данная информация может использоваться при возникновении вопросов после передачи продукта на этап использования в коммерческих целях
Пример	<release_rollout> <sign_off>Mary Jane</sign_off> <sign_off_date>2008-01-16T12:00:00</sign_off_date> <by>John Smith</by> </release_rollout>		

8.4.19 Верификация релиза ('release_verification')

XML-тег	release_verification		
Тип	Комплексный тип		
Определение	Подтверждающая информация о том, что пакет релизов был проверен в тестовом окружении, соответствующем требованиям целевого производственного окружения. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз		
Структура данных	XML-тег	Тип	Определение
	Sign_off	Символьная XML-строка Одно вхождение	В этой записи указывается лицо, утвердившее правильность тестирования программного обеспечения в управляемом окружении и готовность его перехода на этап тестирования в опытном окружении
	Sign_off_ date	XML-тип dateTime Одно вхождение	В этой записи указывается дата утверждения тестирования программного обеспечения
	By	Символьная XML-строка Одно вхождение	В этой записи указывается специалист-практик по использованию процессов SAM, который управлял этапом тестирования продукта в управляемом окружении. Данная информация может использоваться при возникновении вопросов после передачи продукта на этап тестирования в опытном окружении

Окончание таблицы

Пример	<release_verification> <sign_off>Jane Smith</sign_off> <sign_off_date>2008-01-14T12:00:00</sign_off_date> <by>Doug Johnson</by> </release_verification>

8.4.20 Серийный номер ('serial_number')

XML-тег	serial_number
Тип	Символьная XML-строка
Определение	Уникальный идентификационный номер; может быть представлен в виде сочетания цифр, букв или символов. Серийный номер представляет собой общеупотребительный уникальный номер, назначаемый для идентификации конкретного права и покупки. При применении тегов идентификации программного обеспечения основным уникальным ключом становится элемент unique_id, однако многие организации по-прежнему желают, если это возможно, пользоваться серийным номером. Примечания 1 Серийный номер может быть пропущен через одностороннюю хэш-функцию, скрывающую фактический серийный номер – данная особенность по-прежнему полезна для специалистов-практиков по использованию процессов SAM – в особенности если один и тот же ссылочный серийный номер включается в заказ на покупку, счет на оплату или иные сведения, предоставляемые дистрибьютором для потребителя программного обеспечения. 2 Если создатель тега примет решение не предоставлять серийный номер, он может предоставить, например, ряд других ссылочных данных, которые могут использоваться для сопоставления информации в заказах на покупку. Таким образом создатели тегов могут оказывать содействие поставщикам систем SAM в нахождении информации о правах на использование. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз
Пример	<serial_number>1088-9015-2034-4567</serial_number> или <serial_number>10PQR28FTQN2008</serial_number>

8.4.21 SKU ('sku')

XML-тег	sku
Тип	Символьная XML-строка
Определение	Единица складского хранения (Stock Keeping Unit, SKU) представляет собой уникальный идентификационный номер для поставщика программного обеспечения. Номер SKU может быть представлен в виде сочетания цифр, букв или символов. Номер SKU представляет собой общеупотребительный уникальный номер, назначаемый для идентификации конкретного названия и покупки. При применении тегов идентификации программного обеспечения основным уникальным ключом становится элемент unique_id, однако многие организации по-прежнему желают иметь непосредственный доступ к значению номера SKU. Примечание – Если создатель тега примет решение не предоставлять номер SKU, он может предоставить, например, ряд других ссылочных данных, которые могут использоваться для сопоставления информации в заказах на покупку. Таким образом создатели тегов могут оказывать содействие поставщикам систем SAM в нахождении информации о правах на использование. Этот элемент может встречаться в теге идентификации программ-ного обеспечения либо ни разу, либо один раз
Пример	<sku>065-04940</sku>

8.4.22 Алиас создателя программного обеспечения ('software_creator_alias')

XML-тег	software_creator_alias
Тип	Комплексный тип – EntityDataComplexType
Определение	В этом элементе содержится дополнительная информация о создателе программного обеспечения, позволяющая специалистам-практикам по использованию процессов SAM и поставщикам инструментария SAM идентифицировать предыдущие объекты, имевшие отношение с созданию программного обеспечения, идентифицируемого тегом. Несмотря на то, что использование

Окончание таблицы

<table>
<tr><td>Определение</td><td colspan="3">этого элемента не является строго обязательным для процессов обнаружения программного обеспечения, данная запись позволяет облегчить задачу специалистов-практиков по использованию процессов SAM, предоставляя в их распоряжение сведения о предыдущих создателях программного обеспечения, которые можно использовать для поиска более ранних вариантов прав на использование программного обеспечения. Это может быть особенно важно в случае, когда разрешается обновление с версии продукта предыдущего поставщика программного обеспечения на версию продукта текущего поставщика программного обеспечения.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз.</td></tr>
<tr><td rowspan="2">Структура данных</td><td>XML-тег</td><td>Тип</td><td>Определение</td></tr>
<tr><td>alias</td><td>Комплексный тип AliasDetailsComplexType
От ни одного до неограниченного количества вхождений</td><td>Сведения о предыдущих создателях, которые могут иметь отношение к названию программного обеспечения, идентифицированному тегом идентификации программного обеспечения</td></tr>
<tr><td>Пример</td><td colspan="3">Следующий пример относится к продукту Macrovision, который был приобретен и в настоящее время находится в собственности Adobe®
<software_creator_alias>
<alias>
<alias_name>Macrovision</alias_name>
<alias_regid>regid.1998-02.com.macrovision</alias_regid>
</alias>
</software_creator_alias>
или, если regid объекта алиаса неизвестен:
<software_creator_alias>
<alias>
<alias_name>Macrovision</alias_name>
<alias_regid>unknown</alias_regid>
</alias>
</software_creator_alias></td></tr>
</table>

8.4.23 Алиас лицензиара программного обеспечения ('software_licensor_alias')

<table>
<tr><td>XML-тег</td><td colspan="3">software_licensor_alias</td></tr>
<tr><td>Тип</td><td colspan="3">Комплексный тип – EntityDataComplexType</td></tr>
<tr><td>Определение</td><td colspan="3">В этом элементе содержится дополнительная информация о лицензиаре программного обеспечения, позволяющая специалистам-практикам по использованию процессов SAM и поставщикам инструментария SAM идентифицировать предыдущие объекты, имевшие отношение с лицензированию программного обеспечения, идентифицируемого тегом. Несмотря на то, что использование этого элемента не является строго обязательным для процессов обнаружения программного обеспечения, данная запись позволяет облегчить задачу специалистов-практиков по использованию процессов SAM, предоставляя в их распоряжение сведения о предыдущих лицензиарах программного обеспечения, которые можно использовать для поиска более ранних вариантов прав на использование программного обеспечения. Это может быть особенно важно в случае, когда разрешается обновление с версии продукта предыдущего поставщика программного обеспечения на версию продукта текущего поставщика программного обеспечения.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз</td></tr>
<tr><td rowspan="2">Структура данных</td><td>XML-тег</td><td>Тип</td><td>Определение</td></tr>
<tr><td>alias</td><td>Комплексный тип AliasDetailsComplexType
От ни одного до неограниченного количества вхождений</td><td>Сведения о предыдущих лицензиарах, которые могут иметь отношение к названию программного обеспечения, идентифицированному тегом идентификации программного обеспечения</td></tr>
<tr><td>Пример</td><td colspan="3"><software_licensor_alias>
<alias>
<alias_name>Adobe Systems</alias_name>
<alias_regid>regid.1986-12.com.adobe</alias_regid>
</alias>
</software_licensor_alias></td></tr>
</table>

Окончание таблицы

Пример	или, если regid объекта алиаса неизвестен: <software_licensor_alias> <alias> <alias_name>Adobe Systems</alias_name> <alias_regid>unknown</alias_regid> </alias> </software_licensor_alias>

8.4.24 Поддерживаемые языки ('supported_languages')

XML-тег	supported_languages		
Тип	Языки определяются согласно стандарту IETF RFC 4646		
Определение	Языки, которые программный интерфейс представляет для пользователя. Языки должны указываться согласно стандарту IETF RFC 4646. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз		
Структура данных	XML-тег	Тип	Определение
	Язык	Символьная XML-строка От одного до неограниченного количества вхождений	Языки, поддерживаемые данным программным пакетом. Язык может определяться несколько раз. Спецификация языка должна быть определена в соответствии со стандартом IETF RFC 4646 (см. gost.e-ecolog.ru) и процессами, используемыми для сопоставления языковых тегов, как это определено в стандарте IETF RFC 4647 (см. gost.e-ecolog.ru rfc/rfc4647.txt)
Пример	<supported_languages> <language>en</language> <language>fr</language> </supported_languages>		

8.4.25 Алиас создателя тега ('tag_creator_alias')

XML-тег	tag_creator_alias		
Тип	Комплексный тип – EntityDataComplexType		
Определение	В этом элементе содержится дополнительная информация о создателе тега, позволяющая специалистам-практикам по использованию процессов SAM и поставщикам инструментария SAM идентифицировать предыдущие объекты, имевшие отношение к созданию тега идентификации программного обеспечения. Несмотря на то, что использование этого элемента не является строго обязательным для процессов обнаружения программного обеспечения, данная запись позволяет облегчить задачу специалистов-практиков по использованию процессов SAM, предоставляя в их распоряжение сведения о предыдущих создателях тега, которые можно использовать для поиска более ранних вариантов прав на использование программного обеспечения. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз		
Структура данных	XML-тег	Тип	Определение
	alias	Комплексный тип AliasDetailsComplexType От ни одного до неограниченного количества вхождений	Сведения о предыдущих создателях тега, которые могут иметь отношение к названию программного обеспечения, идентифицированному тегом идентификации программного обеспечения
Пример	Следующий пример относится к продукту Macrovision, который был приобретен и в настоящее время находится в собственности Adobe® <tag_creator_alias> <alias> <alias_name>Macrovision</alias_name> <alias_regid>regid.1998-02.com.macrovision</alias_regid> </alias> </tag_creator_alias> или, если regid объекта алиаса неизвестен: <tag_creator_alias> <alias> <alias_name>Macrovision</alias_name>		

Окончание таблицы

Пример	<alias_regid>unknown</alias_regid> </alias> </tag_creator_alias>

8.4.26 Авторское право создателя тега ('tag_creator_copyright')

XML-тег	tag_creator_copyright		
Тип	Символьная XML-строка		
Определение	Данный элемент служит для того, чтобы создатель тега мог указать авторское право на данный конкретный тег. Предполагается, что создатели программного обеспечения разрешат сбор и распространение своих тегов, при условии что содержимое тега, указанное создателем, не будет меняться. Таким образом, поставщики инструментария SAM и другие лица смогут свободно пользоваться тегами идентификации программного обеспечения. Независимая третья сторона, создающая теги, может накладывать другие ограничения на использование и/или последующее распространение данных тегов идентификации программного обеспечения. Более подробная информация об авторских правах приведена в приложении F. Элемент аннотации может встречаться в теге идентификации программного обеспечения несколько раз, однако только один раз для каждого определенного языка. Если язык не указан, в качестве языка устанавливается английский («en»). Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз		
Структура данных	XML-тег	Тип	Определение
	lang	Символьная XML-строка. Представляет собой дополнительный атрибут тега	Язык, на котором составлена аннотация. Языки должны указываться согласно стандарту IETF RFC 4646 - http://gost.e-ecolog.ru rfc/rfc4646.txt
Пример	<tag_creator_copyright lang=«en»>This tag may be used by used, stored, referenced and distributed by any software tool provider and or third party tag collection agency as long as the following elements are not modified: - entitlement_required_indicator - product_title - product_version - software_creator - software_licensor - software_id - tag_creator Extended information may also be added to the tag. </tag_creator_copyright>		

8.4.27 Версия тега ('tag_version')

XML-тег	tag_version
Тип	Комплексный тип
Определение	Этот элемент содержит элемент данных для создателей или модификаторов тега, с помощью которого они могут указывать версию тега. Правильно определенный тег идентификации программного обеспечения не требует наличия версии, указываемой создателем тега, поскольку каждый тег идентификации программного обеспечения является уникальным. Тем не менее, так как в течение жизненного цикла программного обеспечения может происходить перемещение тегов, многим модификаторам тегов может потребоваться вносить определенные изменения в элементы (разрешенные к изменению), и/или добавлять расширенные элементы к тегу идентификации программного обеспечения. В этих случаях требуется наличие ссылки на версию. Необходимо обеспечить, чтобы разные объекты могли указывать собственную информацию о версии, другими словами, данный элемент может быть включен в один тег идентификации программного обеспечения несколько раз. Если в теге присутствует элемент «version» (версия), с целью обеспечения уникальности необходимо указывать все его составляющие. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз

Окончание таблицы

Структура данных	XML-тег	Тип	Определение
	имя	Символьная XML-строка Одно вхождение	Этот элемент указывает имя объекта, определенного в теге. Это имя должно быть согласовано между программными продуктами и релизами программного обеспечения
	regid	тип regid Одно вхождение	Идентификатор regid создателя программного обеспечения (см. 6.1.3.) Если объект неизвестен или больше не занимается бизнесом, это значение может быть установлено в «unknown» (неизвестно)
	numeric_version	ProductVer-sion ComplexTy-pe Комплексный тип, состоящий из четырех элементов, имеющих числовые значения: «major», «minor», «build», «review» Одно вхождение	Числовой идентификатор версии
Пример	<tag_version> <name>My Example Corp</name> <regid>regid.1995-09.com.example</regid> <numeric_version> <major>1</major> <minor>0</minor> <build>0</build> <review>0</review> </numeric_version> </tag_version>		

8.4.28 Обновление для ('upgrade_for')

XML-тег	upgrade_for		
Тип	Комплексный тип		
Определение	Наименование продукта, представляющего собой обновление более ранней версии нижнего уровня, с указанием конкретных данных о том, что именно обновляется. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз		
Структура данных	XML-тег	Тип	Определение
	upgrade_id	Тип идентификации программного обеспечения, описанный в 8.3.5 От одного до неограниченного количества вхождений	Представляет собой ссылку на идентификаторы software_id, которые могут быть обновлены до версии, указанной в данном теге. Если в теге указаны идентификаторы software_id, администраторы систем SAM могут выполнять полностью автоматизированную выверку, гарантирующую выполнение всех обновлений надлежащим образом
	upgrade_description	Символьная XML-строка Ни одного или одно вхождение	Необязательное описание обновления
Пример	<upgrade_for> <upgrade_id> <unique_id>fc3cc419-b5a1-9f16-ed203e537c40</unique_id> <tag_creator_regid>regid.1986-12.com.adobe</tag_creator_regid> </upgrade_id> <upgrade_description>{Optional description of upgrade}</upgrade_description> </upgrade_for>		

8.4.29 Идентификатор использования ('usage_identifier')

<table>
<tr><td>XML-тег</td><td colspan="3">usage_identifier</td></tr>
<tr><td>Тип</td><td colspan="3">комплексный тип</td></tr>
<tr><td>Определение</td><td colspan="3">Содержит информацию, указывающую, какие запущенные процессы должны использоваться для проверки использования продукта. В элементе использования не обязательно должны указываться компоненты программного обеспечения, загружаемые при начальной загрузке, в нем могут быть просто перечислены компоненты, указывающие на то, что конечный пользователь фактически пользуется продуктом.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз.</td></tr>
<tr><td>Структура данных</td><td>XML-тег</td><td>Тип</td><td>Определение</td></tr>
<tr><td></td><td>имя файла</td><td>Символьная XML-строка
От ни одного до неограниченного количества вхождений</td><td>Этот элемент определяет имя файла, исполняемого для запуска приложения.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз</td></tr>
<tr><td></td><td>process-name</td><td>Символьная XML-строка
От ни одного до неограниченного количества вхождений</td><td>Определяет имя процесса в том виде, в котором оно занесено в таблицу процессов для приложения.
По умолчанию данное значение интерпретируется как буквенное выражение. Элемент processname имеет атрибут, называемый «type» (тип), позволяющий определять тип выражения – «literal» (буквенное) или «regexp» (регулярное). Если типом элемента processname является «regexp», то значение должно быть составлено в соответствии с синтаксисом регулярных выражений, определенном в документе http://www.w3.org/TR/xmlschema-2/#regexs.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз</td></tr>
<tr><td></td><td>URI</td><td>XML-тип URI
От ни одного до неограниченного количества вхождений</td><td>Этот элемент определяет расположение в сети веб, к которому необходимо обратиться во время выполнения для определения факта использования приложения. Это значение может использоваться при применении онлайнового инструмента, с помощью которого организация осуществляет мониторинг использования, или в тех случаях, когда во время выполнения приложения требуется получение информации по URI.
По умолчанию данное значение интерпретируется как буквенное выражение. Элемент URI имеет атрибут, называемый «type» (тип), позволяющий определять тип выражения – «literal» (буквенное) или «regexp» (регулярное). Если типом элемента URI является «regexp», то значение должно быть составлено в соответствии с синтаксисом регулярных выражений, определенном в документе xmlschema-2/#regexs.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз</td></tr>
<tr><td>Пример</td><td colspan="3"><usage>
<filename>winword.exe</filename>
<processname>windword.exe</processname>
</usage>
Или
<usage>
<URI type="regexp">https?://[^/]*/MyWebApp/</URI>
</usage></td></tr>
</table>

gost.e-ecolog.ru

8.4.30 Валидация ('validation')

<table>
<tr><td>XML-тег</td><td colspan="3">validation</td></tr>
<tr><td>Тип</td><td colspan="3">Комплексный тип</td></tr>
<tr><td>Определение</td><td colspan="3">Содержит название подпрограммы, которую агент обнаружения может вызвать для проверки (валидации) тега идентификации программного обеспечения. Вследствие системных проблем, ошибок в процессе установки и удаления или по другим причинам, тег идентификации программного обеспечения может быть синхронизирован с установкой программного обеспечения не полностью. Данный элемент позволяет включать в тег идентификации программного обеспечения название подпрограммы валидации, которую при необходимости можно вызвать для проверки правильности тега идентификации программного обеспечения.
Предполагается, что программные приложения осуществляют регулярную проверку тегов идентификации программного обеспечения, с которыми они ассоциированы на момент выполнения – данный процесс можно считать процессом самовосстановления. Во время такого самовосстановления происходит обновление подэлементов тега last_validated_by и last_validated_date, являющихся частью элемента валидации, чтобы следить за тем, что тег прошел процедуру сравнения, и неточностей в нем не обнаружено.
В случаях, когда программный пакет не запускается на выполнение в течение длительного периода времени, обновление подэлементов last_validated_by и last_validated_date не осуществляется. В соответствии с требованиями политики, агент обнаружения может потребовать, чтобы проверка правильности тега идентификации программного обеспечения осуществлялась не реже чем через определенные периоды времени (например, если тег идентификации программного обеспечения не проверялся в течение 3 месяцев, такой тег может считаться подозрительным и, возможно, является не синхронизированным). Если значение last_validated_date не соответствует указанному периоду (старше его), агент обнаружения может вызвать подпрограмму, определенную в элементе validation_call, чтобы выполнить обновление тега идентификации программного обеспечения.
Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо один раз.</td></tr>
<tr><td>Структура данных</td><td>XML-тег</td><td>Тип</td><td>Определение</td></tr>
<tr><td></td><td>Validation_call</td><td>Символьная XML-строка
Одно вхождение</td><td>Представляет собой подпрограмму, которую агент обнаружения может вызвать для проверки правильности (валидации) тега идентификации программного обеспечения. Предполагается, что данный вызов подпрограммы будет осуществляться к одному из исполняемых приложений, содержащихся в программном пакете, с использованием в командной строке параметра, указывающего на необходимость проверки тега на правильность.
Значение validation_call может быть также указано виде ссылки на URL</td></tr>
<tr><td></td><td>last_validated_by</td><td>Символьная XML-строка
Ни одного или одно вхождение</td><td>Этот элемент определяет процесс, который был использован для валидации тега идентификации программного обеспечения. Предполагается, что создатель программного обеспечения создаст ID для всех подпрограмм валидации и включит эти ID в этот элемент.
В зависимости от создателя программного обеспечения, этот ссылочный ID может быть одним и тем же для всех названий программного обеспечения создателя программного обеспечения (например, подпрограмма валидации ACME), или может быть уникальным для каждого пакета. В некоторых случаях для валидации тега может использоваться даже программа установки</td></tr>
<tr><td></td><td>last_validated_date</td><td>XML-тип dateTime
Ни одного или одно вхождение</td><td>Дата и время последней валидации данного тега</td></tr>
<tr><td></td><td>last_validated_result</td><td>Символьная XML-строка
Ни одного или одно вхождение</td><td>Значением данного элемента может быть «true» (истина), «false» (ложь) или «unknown» (неизвестно) (регистр символов строке не имеет значения). Назначением данного элемента является определение метода, с помощью которого можно установить, что, если элемент Validation_call возвратил значение «True», то тег оказался действительным, если элемент возвратил значение «False»,</td></tr>
</table>

Окончание таблицы

Структура данных			то тег оказался недействительным и должен быть проанализирован и изменен, или, если элемент возвратил значение «Unknown», то вызов подпрограммы не мог быть осуществлен (или подпрограмма не возвратила результат), и поэтому действительность тега установить не удалось. Специалист-практик по использованию процессов SAM, получая информацию тега идентификации программного обеспечения, содержащую элемент «Validation_result=false», понимает, что произошла исключительная ситуация, которая требует разрешения. Данная информация не означает, что тег не является действительным, она просто означает, что при выполнении процесса валидации возникла проблема, и специалист-практик по использованию процессов SAM просто уведомляется об этой проблеме
Пример	<validation> <validation_call>c:\program files\ACME\ACME_validator.exe /tag-validate</validation_call> <last_validated_by>ACME_validator.exe</last_validated_by> <last_validated_date>2008-03-31T12:00:00</last_validated_date> <last_validated_result>true</last_validated_result> </validation>		

8.5 Расширенные элементы

8.5.1 Расширенная информация ('extended_information')

XML-тег	extended_information
Тип	Последовательность элементов любого типа
Определение	Вспомогательная информация, которая может предоставляться создателями программного обеспечения или создателями тега, покупателями программного обеспечения или третьими сторонами (например, дистрибьютором, инструментарием SAM или инструментами управления настольными системами). Данные должны быть представлены в формате структуры XML. Данный элемент содержит любую требуемую расширенную информацию. Данные, представленные в настоящем разделе, должны быть оформлены в соответствии с форматом структуры XML. Кроме того, для того чтобы можно было надлежащим образом проверить правильность данных, описанных в настоящем разделе, должен быть представлен XSD-файл. Ссылка на XSD-файл должна быть оформлена в XML-файле тега идентификации программного обеспечения в соответствии со стандартными определениями XML. Поскольку данный элемент не является обязательным, в теге идентификации программного обеспечения его можно не указывать. Если данный элемент присутствует в теге идентификации программного обеспечения, он может встречаться в нем несколько раз. Предполагается, что владельцем каждого раздела extended_information, включенного в тег идентификации программного обеспечения, является одна организация (она же и управляет этим разделом). Таким образом, создатель программного обеспечения может предоставить один элемент extended_information, реселлер может предоставить другой элемент extended_information element, а администратор релизов может предоставить третий элемент extended_information. Каждая из этих записей будет независимой, и модификатор тегов, в общем случае, не должен вносить изменения в данные раздела extended_information, который он не создавал и которым он не владеет. Этот элемент может встречаться в теге идентификации программного обеспечения либо ни разу, либо неограниченное количество раз
Пример	<extended_information> <software_creator_activation_ref>xyzzy</software_creator_activation_ref> </extended_information>

8.6 Определения типов данных

8.6.1 AliasDetailsComplexType

Тип данных	Комплексный тип

Окончание таблицы

Определение	Этот тип используется для определения алиасов, которые ранее могли быть ассоциированы с программным обеспечением, идентифицированным в теге. Алиасы ассоциируются с элементами software_creator_alias, software_licensor_alias и tag_creator_alias. В общем случае алиасы должны использоваться, если организация меняет наименования, или при изменении владельцев программных продуктов. Данные алиасов необходимы специалистам-практикам по использованию процессов SAM для того, чтобы сопоставить название обнаруженного программного обеспечения с ранее приобретенными версиями связанного программного обеспечения.		
Структура данных	XML-тег	Тип	Определение
	alias_name	Символьная XML-строка Одно вхождение	Этот элемент содержит определение имен предыдущих объектов, которые могли быть ассоциированы с программным обеспечением, определенным в конкретном теге
	alias_regid	Символьная XML-строка Одно вхождение	Этот элемент содержит определение указанных идентификаторов Regid предыдущих объектов (как это определено в 6.1.3). Если объект неизвестен или больше не занимается бизнесом, это значение может быть установлено в «unknown» (неизвестно)
Пример	<alias> <alias_name>Macrovision</alias_name> <alias_regid>regid.1998-02.com.macrovision</alias_regid> </alias>		

8.6.2 EntityComplexType

Тип данных	Комплексный тип		
Определение	Этот тип используется для определения конкретных уникальных сведений об объектах, определенных в элементах software_creator, software_licensor или tag_creator		
Структура данных	XML-тег	Тип	Определение
	имя	Символьная XML-строка Одно вхождение	Этот элемент указывает имя объекта, определенного в теге. Это имя должно быть согласовано между программными продуктами и релизами программного обеспечения
	regid	Символьная XML-строка Одно вхождение	Идентификатор Regid объекта. Если объект неизвестен или больше не занимается бизнесом, это значение может быть установлено в «unknown» (неизвестно)
Пример	<tag_creator> <name>Adobe</name> <regid>regid.1986-12.com.adobe</regid> </tag_creator>		

8.6.3 EntityDataComplexType

Тип данных	Комплексный тип		
Определение	Этот тип используется для определения дополнительных алиасов для объектов, ассоциированных с программным обеспечением, идентифицированным в конкретном теге идентификации программного обеспечения. Такими алиасами могут быть любые алиасы (другие владельцы), которые ранее могли быть ассоциированы с программным обеспечением, идентифицированным в теге, а также идентификатор regid, указываемый для конкретного объекта		
Структура данных	XML-тег	Тип	Определение
	alias	Комплексный тип AliasDetailsComplexType От ни одного до неограниченного количества вхождений	Этот элемент содержит определение предыдущих объектов, которые могли быть ассоциированы с программным обеспечением, определенным в конкретном теге. В общем случае алиасы должны использоваться, если организация меняет наименования, или при изменении владельцев программных продуктов. Данные алиасов необходимы специалистам-практикам по использованию процессов SAM для того, чтобы сопоставить название обнаруженного программного обеспечения с ранее приобретенными версиями связанного программного обеспечения

Окончание таблицы

Пример	Следующий пример относится к продукту Macrovision, который был приобретен и в настоящее время находится в собственности Adobe® <tag_creator_alias> <alias> <alias_name>Macrovision</alias_name> <alias_regid>regid.1998-02.com.macrovision</alias_regid> </alias> </tag_creator_alias>

8.6.4 GUIDType

Тип данных	тип GUID	
Определение	Этот тип данных определяет GUID и позволяет выполнять проверку того, что значения данных соответствуют определению GUID, которое может выглядеть следующим образом: 00001101-0000-1000-8000-00805f9b34fb	
Структура данных	Тип	Строковый
	Ограничения	значение шаблона="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}"

8.6.5 ProductVersionComplexType

Тип данных	Комплексный тип		
Определение	Этот тип данных создается для представления стандартизованной 4-значной числовой версии. Если номер версии, используемый для продукта, имеет менее 4 уровней, нижние уровни должны быть установлены в значение '0'		
Структура данных	XML-тег	Тип	Определение
	major	Целое Одно вхождение	Самый высокий уровень номера версии. Обычно этот уровень называют основным номером версии
	minor	Целое Одно вхождение	Второй уровень номера версии. Обычно этот уровень называют дополнительным номером версии
	build	Целое Одно вхождение	Третий уровень номера версии. Многие организации называют этот уровень сборочной версией
	review	Целое Одно вхождение	Четвертый уровень номера версии. Многие организации называют этот уровень патчевым, или корректирующим уровнем

8.6.6 FootprintModuleComplexType

Тип данных	Комплексный тип		
Определение	Этот тип используется для определения файлов, записей реестра и других значений данных, ассоциированных с конкретной установкой программного пакета		
Структура данных	XML-тег	Тип	Определение
	referenced	XML-тип URI Ни одной или одна запись	Этот элемент аналогичен элементу external_description, однако может использоваться только для конкретного модуля «отпечатка»
	file	Комплексный тип footprint file <file> < name> (<size>)* (<md5>)* (<version>)* (<other name>)*</file> От ни одной до неограниченного количества записей	Этот элемент представляет собой описание файла с его атрибутами: размером, MD5, версией и любой другой необходимой информацией

Продолжение таблицы

Структура данных	os_configu-ration_re-cord	Комплексный тип конфигурационной записи ОС <os_configura-tion_record> <record_type> (<path>)? (<name>)? (<internal_path>)? (<entry>)+ (<name>)? (<value>)? (<type>)? </entry> </os_configura-tion_re-cord> От ни одной до неограниченного количества записей	Этот элемент выдает конфигурационную информацию операционной системы, которую производитель программного обеспечения может пожелать предоставить, чтобы определить факт установки его программного обеспечения. Возможными значениями элемента record_type могут быть следующие: a) Registry (Реестр) b) WMI c) RPM d) ODM e) file-entry (файловая запись) Ниже приведены примеры использования этих составляющих. Со временем могут быть определены другие типы; информация об обновлении типов приводится на вебстранице ISO 19770 – gost.e-ecolog.ru 9770/
	other	Комплексный тип footprint other <other type> <param name> </other> От ни одной до неограничен-ного количества записей	Этот элемент содержит всю требуемую информацию об «отпечатке», не включенную в перечисленный выше список. Этот элемент может содержать любые атрибуты
Пример	Примеры записей <os_configuration_record> registry An os_configuration_record with record_type of registry uses the specified elements as a regular request to capture registry values from Microsoft® Windows® computers. Elements are used as follows: <record_type> registry <path> full path to the registry value. This includes the fully specified root (i.e. HKEY_LOCAL_MACHINE) <name> NA <internal_path> NA <entry> <name> registry value required for comparison <value> comparison value <type> type of registry value </entry> Example: <os_configuration_record> <record_type>registry</record_type> <path> HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Reader\7.0\AdobeViewer</path> <entry> <name>EULA</name> <value>1</value> <type>REG_DWORD</type> </entry> </os_configuration_record> WMI An os_configuration_record with a record_type of WMI uses the specified elements as part of a WBEM Query Language (WQL) request. Elements are used as follows: <record_type> WMI <path> namespace where the class is located <name> class of the object <internal_path> query statement used (i.e. where state='stopped' <entry> <name> attribute name		

Продолжение таблицы

Пример	<value> comparison value <type> type of class attribute value (optional) </entry> Example: <os_configuration_record> <record_type>WMI</record_type> <path>Root\CIMV2</path> <name>Win32_Product</name> <internal_path>Name= 'Adobe Acrobat 8 Professional'</internal_path> <entry> </entry> <entry> </entry> <name>Version</name> <value>8.1.2</value> <name>Vendor</name> <value>Adobe Systems</value> <type>string</type> </os_configuration_record> RPM An os_configuration_record with a record type of RPM uses the specified elements to read data from the RPM data store. Elements are used as follows: <record_type> RPM <path> NA <name> name of the RPM package <internal_path> NA <entry> <name> attribute from the RPM package to use for comparison <value> comparison value <type> type of attribute value </entry> Example: <os_configuration_record> <name>lvm2-2.02.28-1.fc8</name> <type>RPM</type> <entry> <name>Name</name> <value>lvm2</value> </entry> <entry> <name>Version</name> <value>2.02.28</value> </entry> <entry> <name>Signature</name> <value>DSA/SHA1, Thu 25 Oct 2007 06:10:53 AM CEST, Key ID b44269d04f2a6fd2</value> <type>string</type> </entry> <entry> <name>Packager</name> <value>Fedora Project</value> </entry> </os_configuration_record> ODM
	An os_configuration_record with a record type of ODM uses the specified elements to read data from the AIX Object Data Manager. Elements are used as follows: <record_type> ODM <path> NA <name> type of ODM information to retrieve

Окончание таблицы

Пример	<internal_path> NA <entry> <name> attribute from the LPP package to use for comparison <value> comparison value <type> type of attribute value </entry> Example: <os_configuration_record> <type>ODM</type> <name>lpp</name> <entry> <name>name</name> <value>bos.msg.en_US.docsearch.client.Dt</value> <type>string</type> </entry> <entry> <name>description</name> <value>Lite NetQuestion Local Web Server</value> <type>string</type> </entry> <entry> <name>ver</name> <value>5</value> <type>short</type> </entry> </os_configuration_record> File_entry <record_type> file_entry <path> path to file <name> NA <internal_path> NA <entry> <name> NA <value> comparison value to find in file <type> type of attribute value </entry> Example: <os_configuration_record> <type>file-entry</type> <path>/etc/inittab</path> <entry> <value>x:5:respawn:/etc/X11/prefdm -nodaemon</value> </entry> </os_configuration_record>

Приложение А
(справочное)

Принципы создания и ведения тегов идентификации программного обеспечения

А.1 Введение

Назначением настоящего приложения является представление общего обзора принципов создания и ведения тегов идентификации программного обеспечения. В Приложении приводятся ссылки на множество нормативных требований, определенных в основном тексте данной части настоящего стандарта, однако без добавления или изменения этих требований.

Главной целью тега идентификации программного обеспечения является упрощение процессов идентификации и управления установленным программным обеспечением. Данные, содержащиеся в теге идентификации программного обеспечения, должны учитываться при создании, распространении, лицензировании и использовании программного обеспечения.

А.2 Жизненный цикл тега идентификации программного обеспечения

А.2.1 Обзор

Данные, содержащиеся в теге идентификации программного обеспечения, создаются и/или изменяются в четырех основных точках жизненного цикла тега идентификации программного обеспечения, как это показано на следующем рисунке Рис. А.1



Рисунок А.1 – Жизненный цикл тега идентификации программного обеспечения

Каждый этап данного процесса может определяться различными объектами. В общем случае процесс создания определяется создателем программного обеспечения, процесс выпуска релизов определяется администратором релизов, а процесс установки обычно определяется лицензиаром программного обеспечения или создателем программного обеспечения, причем организация потребителя программного обеспечения может добавлять в тег определенную информацию, относящуюся к установке.

А.2.2 Процесс создания

После создания программного обеспечения в виде исходной эталонной копии в это программное обеспечение часто включается тег идентификации программного обеспечения, содержащий ряд предопределенных элементов. Владельцами этих элементов, в общем случае, являются создатель программного обеспечения или лицензиар программного обеспечения. К этим элементам относятся следующие (элементы, выделенные жирным шрифтом, являются обязательными):

1. abstract
2. component_of
3. complex_of
4. data_source
5. dependency
6. elements_owner
7. entitlement_required_indicator
8. license_linkage
9. package_footprint

10. product_category
11. product_identifier
12. product_title
13. product_version
14. release_date
15. sku
16. software_creator
17. software_creator_alias
18. software_licensor
19. software_licensor_alias
20. software_id
21. supported_languages
22. tag_creator
23. tag_creator_alias
24. tag_creator_copyright
25. upgrade_for
26. usage_identifier
27. validation

В общем случае эти элементы должны быть относительно статичны для определенного программного продукта и должны быть включены в установочную эталонную копию.

Создатели программного обеспечения могут создавать теги только в процессе установки. Процесс создания тегов будет считаться соответствующим требованиям, если данные, содержащиеся в фактически установленном теге, соответствуют одному и тому же типу данных, которые содержались бы в продукте с предустановочной версией тега. Должны быть определены процедуры, позволяющие потребителю программного обеспечения указывать в теге собственные значения для элементов релиза; упаковщикам программного обеспечения также должна быть предоставлена возможность вносить изменения в элементы тега.

А.2.3 Процесс упаковки программного обеспечения

Некоторое программное обеспечение подвергается процессу упаковки, который может выполняться третьей стороной. Упаковываться могут OEM-продукты, интегрируемые в комплексное программное решение. Упаковку могут выполнять лицензированные упаковщики, объединяющие несколько продуктов в набор.

В этих случаях обновляемые элементы тега могут быть разными и задаваться на основе договоренности между собой создателя программного обеспечения, лицензиара программного обеспечения и упаковщика программного обеспечения.

Упаковщик программного обеспечения, например, может создавать собственный тег идентификации программного обеспечения взамен существующего, или, как вариант, создавать тег пакета, в котором в качестве компонента будет присутствовать ссылка на существующий тег и программное обеспечение. Еще одним вариантом может быть внесение изменений и/или добавлений в элементы тега идентификации программного обеспечения. Если тег просто внедряется в соответствующий продукт без изменения элементов, владельцами которых являются создатель программного обеспечения (элемент software_creator) или лицензиар программного обеспечения (элемент software_licensor), то к элементам, которые могут изменяться или добавляться, относятся следующие (ни один из них не является обязательным):

1. component_of
2. complex_of
3. data_source
4. dependency
5. elements_owner
6. license_linkage
7. packager

Дополнительная информация в перечисленных выше элементах тега идентификации программного обеспечения, предоставляемая упаковщиками программного обеспечения, может использоваться специалистами-практиками по использованию процессов SAM для идентификации программного обеспечения, связанного с конкретным (а не с каким-либо другим) пакетом.

А.2.4 Процесс выпуска релиза

После того как эталонная версия программного обеспечения будет поставлена в организацию, к этой версии часто добавляются специальные данные, относящиеся к установке, после чего выполняется тестирование версии и затем выполняется ее распространение и окончательная установка. На этом этапе владельцем определенных элементов тега является администратор релизов, который может регулярно вносить изменения в эти элементы тега идентификации программного обеспечения. К элементам, относящимся к процессу выпуска релизов, относятся следующие (ни один из них не является обязательным)

1. elements_owner
2. packager
3. release_id
4. release_package
5. release_rollout
6. release_verification

Более крупные по размерам потребительские организации также могут пожелать дополнить тег идентификации программного обеспечения расширенной информацией, содержащей дополнительные данные, которые можно использовать для целей поддержки, выполнения процедур SAM или иных процессов.

А.2.5 Процесс установки

Данный процесс обычно определяется создателем программного обеспечения или лицензиаром программного обеспечения и может быть соответствующим образом изменен администратором релиза внутри организации.

При установке программного продукта на вычислительное устройство тег идентификации программного обеспечения получает окончательное имя файла. Как было указано в 6.1.7, при создании имени файла настоятельно рекомендуется использовать элемент unique_sequence_id, включающий идентифицирующую информацию о машине, так как он придает общую уникальность имени файла в рамках всей организации, но, что более важно, он позволяет идентифицировать систему, которая использовалась при установке программного пакета на съемное устройство.

В процессе установки также происходит обновление ряда значений данных. Такими значениями обычно являются значения следующих элементов (ни один из них не является обязательным).

1. installation_details
2. serial_number
3. validation

А.3 Уникальное определение идентификатора software_id

Уникальный идентификатор software_id соответствует уникальному продукту на двоичном уровне для целей распространения и/или обновления. Уникальность гарантируется сочетанием уникального имени создателя тега (элемент tag_creator_regid) и заведенного создателем тега (tag_creator) идентификатора unique_id.

Идентификатор software_id для конкретной версии конкретного программного пакета должен оставаться согласованным для каждого процесса распространения данного программного пакета. Другие сведения, присутствующие в теге идентификации программного обеспечения, могут меняться, отражая различия в каналах распространения, или даже то обстоятельство, что программный продукт включен комплектную версию программного набора третьей стороны.

Могут встречаться случаи, когда фактический идентификатор software_id набора программного обеспечения неизвестен вплоть до момента установки. Такая ситуация может быть характерна для продуктов, для которых одной программой установщиком можно настроить несколько конфигураций программного обеспечения, и о том, какая именно версия программного пакета будет размещена на вычислительном устройстве, программе-установщику становится известно только в процессе установки. В этих случаях каждый конфигурационный параметр, который может включать различные лицензионные права на на использование, должен сопровождаться собственным уникальным идентификатором software_id.

А.4 Спецификация имени файла

А.4.1 Обзор

Имена файлов имеют две различных формы – одна форма для файлов, имена которых используются до установки программного пакета (дистрибуционное имя файла), другая форма используется при установке программного пакета на вычислительное устройство (установочное имя файла).

А.4.2 Дистрибуционное имя файла

Дистрибуционное имя файла должно создаваться в соответствии с правилами, установленными в 6.1.6. Имя файла включает уникальную идентификационную информацию о программном обеспечении и содержит информацию о создателе тега (элемент tag_creator). Дистрибуционное имя файла указывается создателем программного обеспечения (элемент software_creator) или создателем тега (элемент tag_creator), и, вероятнее всего, будет совершенно идентичным для всех экземпляров для распространения созданного программного обеспечения.

А.4.3 Установочное имя файла

При установке тега идентификации программного обеспечения на вычислительное устройство имя файла должно быть уникальным для этого конкретного устройства. Выполнение этого требования является обязатель-

ным, поскольку в одном каталоге могут размещаться несколько тегов идентификации программного обеспечения, и каждое имя файла должно быть уникальным. Для выполнения этого требования используется спецификация имени файла, определенная в 6.1.7.

Кроме того, настоятельно рекомендуется, чтобы программы установки тегов идентификации программного обеспечения следовали рекомендациям, определенным в 6.1.7, оговаривающим включение в установочное имя файла уникальной информации о машине. Это позволит специалистам-практикам по использованию процессов SAM (и другим лицам) идентифицировать машину, использованную для установки программного обеспечения на съемный или используемый совместно (сетевой) носитель.

Уникальная информация о машине может также включать сведения, относящиеся к конкретной виртуальной машине. Несмотря на то, что такая информация в 6.1.7 специально не идентифицируется, чем более уникальной будет информация, содержащаяся в ID машины, тем с большей вероятностью эти теги идентификации программного обеспечения могут быть ассоциированы с конкретным устройством, которое было использовано для установки программного обеспечения.

А.5 Каталоги для установки тегов

Тег идентификации программного обеспечения может устанавливаться на вычислительное устройство в два места. Первое место – общий системный каталог (см. 6.1 и А.6.3); второе место – каталог верхнего уровня установочного каталога программного пакета.

А.6 Принципы работы без привлечения регистрирующих органов

А.6.1 Общие положения

Данная часть настоящего стандарта составлена для того, чтобы избежать необходимости в привлечении регистрирующих органов. Регистрирующий орган мог бы вести общие списки многих типов информации, охватываемой данной частью настоящего стандарта, например:

a) информацию о всех платформах, их соответствующих владельцах, информацию о том, в каких местах платформы должны храниться теги идентификации программного обеспечения;

b) уникальные имена и идентификаторы создателя программного обеспечения;

c) уникальные идентификаторы программного обеспечения.

А.6.2 Уникальные ссылки на идентифицированных создателей и лицензиаров

В целью упрощения работы без привлечения регистрирующего органа в данную часть настоящего стандарта включены следующие принципы:

a) каждый из элементов elements tag_creator, software_creator и software_licensor должен использовать специальный регистрационный идентификатор (regid), являющийся гарантированно уникальным в пределах организации, и который можно использовать для идентификации организации. Идентификатор regid создается на базе определения, разработанного для стандарта iSCSI, как это определено в стандарте IETF RFC 3720;

b) идентификатор regid включает в себя доменное имя создателя (в соответствии с определением стандарта IETF RFC 1034, 3.5 и стандарта IETF RFC 1123, 2.1). Использование в данной части настоящего стандарта идентификатора regid (и, в расширительном смысле, компонентов домена организации) позволяет указывать уникальный ID, не требующий привлечения независимого регистрирующего органа и снабдить заинтересованных лиц дополнительной информацией для обратного отслеживания программного обеспечения вплоть до создателя первоначального тега;

c) создатель тега несет ответственность за предоставление элементов unique_id по всем данным, создаваемым для идентификатора regid создателя тега;

d) В отношении онлайновой ссылочной информации, например, для информации об «отпечатке» пакета, устанавливаются специальные условия. Если данная информация используется в качестве онлайновой ссылки, необходимо указание идентификатора URI, уникально идентифицирующего создателя тега.

Такой подход позволяет создателям тега существовать и работать независимо от создателей программного обеспечения, что позволяет создавать теги идентификации программного обеспечения для программного обеспечения создателей программного обеспечения, которые, возможно, уже отошли от бизнеса, или которые не создают теги для своего программного обеспечения. Такой подход также позволяет создавать теги идентификации программного обеспечения для программного обеспечения, созданного до выпуска данной части настоящего стандарта.

А.6.3 Данные о каталогах хранения платформы

Единственной информацией, которая не содержится в самом теге, или по встроенной в него ссылке, являются данные о том, в каком месте данной платформы (например, Windows®, UNIX® или Linux™) будут храниться теги идентификации программного обеспечения. Количество платформ сравнительно невелико, и требования к этой информации должны обрабатываться примерно одинаковыми способами, а именно:

a) каждый провайдер платформы вправе указать, в каком месте его платформы будет храниться данная информация. Провайдер платформы должен иметь возможность сообщать данные сведения любым выбранным

им способом, например, через свой веб-сайт. Данная часть настоящего стандарта рекомендует, чтобы провайдеры платформы, как минимум, публиковали данную информацию в подкаталоге с именем «19770» на странице с основным адресом доменного имени;

b) общие системные каталоги, которые должны использоваться для хранения тегов на различных платформах, были определены в 6.1;

c) технические отчеты могут публиковаться по адресу: gost.e-ecolog.ru и содержать сводную информацию об известных платформах и правилах оформления значений данных, наиболее часто используемых при работе с тегами идентификации программного обеспечения.

Далее, каждый создатель программного обеспечения должен иметь уникальный идентификатор создателя программного обеспечения (элемент 'software_creator'), который позволяет упростить процесс объединения информации создателем программного обеспечения, даже если такая информация была создана различными создателями тегов. С другой стороны, отсутствие таких уникальных идентификаторов для конкретных создателей программного обеспечения, никоим образом не влияет на успешную реализацию данной части настоящего стандарта.

Приложение В
(справочное)

Сценарии действий и инструкции для поставщиков программного обеспечения

В.1 Введение

Поставщик программного обеспечения может реализовывать настоящий стандарт по нескольким причинам:

a) Простота идентификации

Потребителям программного обеспечения станет легче идентифицировать программное обеспечение и осуществлять его инвентаризацию; что позволит расширить использование практических методов SAM. Аудит программного обеспечения и аудиторы программного обеспечения (внутренние или иные) получат более глубокое понимание того, что именно установлено в организации.

b) Точность идентификации

Существующие методы идентификации программного обеспечения обычно базируются на программных методах распознавания подписей программных компонентов, установленных на машинах. Эти подписи часто относятся не к тому же уровню разрешений, что и уровень прав на использование программного обеспечения. Кроме того, многие наименования продуктов не имеют явной связи с фактически установленными компонентами приложений. Эти проблемы приводят к сложности процессов выверки результатов идентификации с правами на использование программного обеспечения.

c) Контроль над процессами идентификации программного обеспечения

С целью обеспечения целостности, создатели программного обеспечения имеют возможность точно указать, что можно, а что нельзя изменять в теге идентификации программного обеспечения (далее – тег).

После того как будут стандартизованы электронные права на использование программного обеспечения, создатель программного обеспечения и провайдер, который уже использует теги, смогут реализовывать права на использование программного обеспечения, обеспечивающие их автоматизированную или практически автоматизированную выверку с существующими тегами. По мере того как создатели программного обеспечения будут внедрять теги в линейки своих продуктов, им потребуется учитывать то обстоятельство, что права на использование программного обеспечения могут повлиять на объемы информации, содержащейся в теге.

С точки зрения определения программного продукта и с точки зрения развития лучше всего определять тег на самых ранних стадиях проекта, чтобы программная часть проекта выполнялась с использованием правильных инструментов и технологий, обеспечивающих соблюдение требований, оговоренных для целевых языков и платформ.

Использование стандартизованных тегов одинаково выгодно как для потребителей программного обеспечения, так и для его создателей. Потребители программного обеспечения могут повысить эффективность своей работы за счет использования упрощенных процессов обнаружения и повышения общей эффективности процессов. Выгода потребителей программного обеспечения, у которых появятся надежные средства управления программными активами, обеспечивающие установку и использование программного обеспечения в соответствии с соглашениями о предоставлении лицензий на программное обеспечение, также скажется и на создателях программного обеспечения.

Ниже приводятся сценарии действий, отражающие различные подходы к созданию и обновлению тега в течение жизненного цикла программного продукта.

В.2 Роли, задействованные в процессе создания/изменения тега идентификации программного обеспечения

За правильность процессов создания и управления тегом идентификации программного обеспечения несут многие лица с различными должностными функциями, в том числе (но не ограничиваясь следующим списком):

a) менеджер по продукту – это лицо определяет разрабатываемый/усовершенствуемый продукт и определяет множество аспектов продукта, которые должны быть отражены в теге. Использование для определения продукта информации, представленной в теге, позволят создавать более подробную документацию по продукту;

b) руководитель разработки – это лицо определяет технологию, используемую для разработки и поставки программного обеспечения.

Предварительно определенные спецификации тега позволят им более глубоко понимать окружение, в котором конечные пользователи будут использовать продукт;

c) специалист по лицензированию программного обеспечения/релизам продуктов – это лицо, предоставляющее конечным пользователям, IT-специалистам и аудиторам информацию о том, какую именно часть программного обеспечения они используют. Если комплект или набор состоит из нескольких продуктов, эта группа лиц должна понимать, как это может повлиять на лицензирование продукта или действия по релизу продукта (в том

числе на обновления каталогов). Это лицо отвечает за обеспечение правильности и надлежащее сопровождение перекрестными ссылками элемента программного обеспечения, его лицензии и любой связанной информации о каталогах.

B.3 Роль менеджера по продукту

Менеджер по продукту отвечает за определение требований к программному продукту. Часть этих требований заключается в определении элементов, которые должны быть включены в тег идентификации программного обеспечения (далее – тег), а также во многих случаях – значений, которые должны быть определены для конкретных элементов тега. Чем больше информации о различных тегах, которые могут быть ассоциированы с продуктом, укажет менеджер по продукту, тем больше данных будет у специалистов по разработке и выпуску релизов программного обеспечения для понимания требований к продукту.

Для менеджера по продукту наибольшую важность могут представлять следующие элементы (которые он должен хорошо понимать).

a) Обязательные элементы

При формировании менеджером по продукту определения продукта шаблон определения должен включать раздел, определяющий требуемые части тега. Сначала менеджер по продукту работает только с обязательными элементами тега:

1) entitlement required (необходима выверка прав на использование) – этот тег определяет, должно ли учитываться обнаруженное программное обеспечение в ходе процесса выверки прав на использование. Менеджер по продукту может четко указать случаи, когда выверка прав на использование необходима (когда лицензия на программное обеспечение продается потребителям программного обеспечения), а когда нет (если программное обеспечение устанавливается в режиме пробного использования или предоставляется бесплатно);

2) product title (наименование продукта) – этот элемент соответствует официальному рыночному наименованию продукта, определенному создателем программного обеспечения. Обычно этим именем является имя, которым специалист-практик по использованию процессов SAM и IT-специалисты называют продукт. Само по себе наименование не должно оказывать непосредственное воздействие на процессы выверки;

3) product version (версия продукта) – этот важный элемент позволяет определять как маркетинговую (текстовую) версию продукта (обычно используемую создателями программного обеспечения для упрощения именования конкретной версии с рекламными целями), так и официальную числовую версию программного обеспечения. Числовая версия может включать до четырех элементов, определяющих полную информацию о версии: основной номер версии, дополнительный номер версии, сборочный номер версии и корректирующий номер версии. Поставщик программного обеспечения может лицензировать программный продукт только по основной или дополнительный версии. В этих случаях менеджер по продукту должен убедиться в том, что информация о правах на использование программного обеспечения (заказ на покупку, счет, сертификат лицензии на программное обеспечение или его эквивалент) могут быть четко привязаны к конкретному номеру версии, чтобы во время выполнения процесса выверки не происходила путаница;

4) software creator identity (идентификационные данные создателя программного обеспечения) – назначение этого элемента состоит в уникальной и согласованной идентификации поставщика, изготовившего программное обеспечение. Учитывая, что названия некоторых региональных отделений программных компаний могут несколько отличаться друг от друга, важно добавить к имени идентификатор, который будет оставаться постоянным во всех странах, регионах и на всех языках – эта информация указывается в элементе regid. Это значение должно быть одинаковым для всех продуктов и релизов, созданных создателем программного обеспечения;

5) software unique identifier (уникальный идентификатор программного обеспечения) – менеджер по продукту совместно с остальными лицами, представляющими организацию-разработчика, определяет уникальный идентификатор для каждой версии продукта. Уникальный идентификатор позволяет обеспечивать надлежащее сравнение в процессе выверки.

b) Дополнительные элементы

Менеджер по продукту определяет другие элементы, требующие определения. Одним их лучших практических способов определения тега идентификации программного обеспечения состоит в следующем: создатель программного обеспечения должен понять, какие дополнительные элементы на момент отгрузки программного обеспечения определены «наилучшим образом» и последующему изменению не подлежат, а в какие программные элементы, возможно, придется вносить изменения по мере того как программное обеспечение проходит по канала продажи и доходит в итоге до места установки. В настоящий стандарт включен исчерпывающий набор дополнительных элементов, дополняющих содержимое тега и повышающих эффективность процессов идентификации и выверки. В этом разделе приводится описание использования нескольких важных дополнительных тегов, определенных в стандарте:

1) component association (ассоциация с компонентами) и components list (список компонентов) – эти два дополнительных элемента позволяют менеджерам по продукту назначать тег программному продукту как компоненту объекта лицензирования продукта, например, набору или комплекту. Аналогично этому, элемент components list (список компонентов) предоставляет возможность формировать перечень оставшихся компонентов, связанных с одним и тем же объектом лицензирования (т. е. с набором). Включение одного или обоих дополнительных элемен-

тов в тег позволяет значительно повысить вероятность правильной идентификации, казалось бы, независимых установок программного обеспечения и одновременно гарантировать, что процесс выверки будет учитывать надлежащую лицензию на программное обеспечение набора по отношению к точечным продуктам, и наоборот;

2) license and channel information (информация о лицензии и канале) – этот дополнительный элемент еще более упрощает идентификацию программного обеспечения, установленного на данной машине, повышая эффективность работы специалиста-практика по использованию процессов SAM и эффективность всего процесса выверки лицензионных прав. Например, зная источник установки, специалисты-практики по использованию процессов SAM смогут с легкостью отделять программное обеспечение, приобретенное непосредственно, от OEM-продуктов, включаемых в новые приобретаемые персональные компьютеры (ПК);

3) package footprint («отпечаток» пакета) – этот элемент позволяет создателю программного обеспечения указывать файлы, записи реестра и другую информацию, которая может быть использована для идентификации программного пакета. Этот элемент разрешает размещение в файле нескольких записей, поэтому в одном ссылочном файле могут обрабатываться и патчи, и незначительные релизы программного обеспечения. Цель этого элемента состоит в предоставлении информации, которую агент обнаружения может использовать для проверки правильности идентификации тегом установленного программного обеспечения. Дополнительным преимуществом является то, что этот элемент позволяет инструментарию для обнаружения тегов исключать из списка обнаруженных файлов все «известные» файлы. Имея достоверный список файлов, записей реестра, записей WMI, а также другую информацию, относящуюся к платформе, по конкретному продукту, инструментарий для обнаружения тегов и специалисты-практики по использованию процессов SAM могут отфильтровывать информацию из списка всех обнаруженных элементов. Отфильтровывая «известную информацию», специалисты-практики, получают точную картину неизвестного или нового программного обеспечения, которое может быть установлено в их рабочем окружении;

4) product identifier (идентификатор продукта) — этот элемент представляет собой идентификатор, сопровождающий конкретный продукт от релиза к релизу. Этот элемент не должен использоваться в качестве маркетингового элемента продукта (как, например, наименование продукта), это должен быть просто идентификатор, не меняющийся от релиза к релизу. Например, организация может создавать и продавать продукт, называемый «Acme Widgets 2007 Pro». Если следующий релиз этого продукта выйдет под другим названием, к примеру, «Acme Widgets 2008 Expert», то специалист-практик, имея только наименование продукта, не будет знать, подпадают ли эти два продукта под действие конкретного соглашения об обслуживании. Тем не менее, если оба продукта имеют идентификатор продукта, например, «fc3cc419-b5a1-9f16- ed203e537c40», то специалист-практик может определить, что оба продукта подпадают под действие одного и того же соглашения об обслуживании, и затем установить их соответствие. Этот элемент позволяет организации, не идентифицируя конкретное наименование, указывать, какие обновления разрешаются осуществлять в рамках соглашений об обслуживании;

5) serial number (серийный номер) – необходимость включения серийного номера в состав тега прямо пропорциональна важности серийного номера как элемента прав на использование программного обеспечения для лицензии на программное обеспечение, приобретаемой потребителями программного обеспечения. Издателям программного обеспечения, требующим ввода серийного номера, привязанного к потребителю программного обеспечения и/или конкретной покупке, для установки и использования программного обеспечения, настоятельно рекомендуется включить в тег элемент serial number (серийный номер) для установления прямой взаимосвязи с заказом на покупку и/или правами на использование программного обеспечения в процессе выверки. Серийные номера известны на момент установки, поэтому между установкой и лицензированием прав на использование программного обеспечения можно определить прямую связь.

i) специальный комментарий по серийным номерам в теге: поскольку серийные номера часто связывают с активированием конкретных функций программного обеспечения, менеджеры по продукту должны решить, включать ли серийный номер в тег идентификации в «чистом виде», либо применить односторонний криптографический хэш к серийному номеру программного обеспечения, привязанному к конкретному потребителю, чтобы повысить защищенность этих данных и снизить риск возможной утечки действующих серийных номеров через сеть Интернет. Если поставщик программного обеспечения решит включить в тег скрытую/хэшированную версию серийного номера, такая же скрытая/хэшированная версия должна быть включена в заказ на покупку, права на использование программного обеспечения или сертификат лицензии на программное обеспечение для использования во время ручного или автоматизированного процесса выверки;

6) SKU – SKU может быть важным элементом для тех компаний, которые не используют серийные номера при активации программного обеспечения. При использовании SKU обычно требуется ассоциировать установленное программное обеспечение с правами на использование программного обеспечения;

7) supported languages (поддерживаемые языки) – элемент supported languages (поддерживаемые языки) позволяет создателям программного обеспечения указывать конкретный язык(языки), установленный на машине. Эта информация представляет важность для поставщиков программного обеспечения, продающих лицензии на программные обеспечение, привязанные к языку, поскольку в правах на использование программного обеспечения/заказе на покупку должны указываться продаваемые потребителю программного обеспечения продукты, привязанные к конкретному языку;

8) software creator alias (алиас создателя программного обеспечения) – этот дополнительный элемент отражает имя создателя, использовавшееся до получения текущего имени; данный элемент очень важен в том случае,

если разрешается обновление от версии предыдущего создателя до версии текущего создателя. Этот элемент следует включать в тег при выпуске новых версий программных продуктов после приобретения;

9) upgrade for (обновление для) – этот элемент разработан для упрощения автоматизированных процессов выверки и обеспечивает правильную выверку обновлений. Этот элемент позволяет продукту идентифицировать самого себя как обновление конкретного продукта или продуктов. Используя данную информацию, специалисты-практики по использованию процессов SAM или аудиторы смогут осуществлять выверку существующих установок обновленного продукта с известными правами на использование старых продуктов. Учитывая, что создатели программного обеспечения будут обновлять лицензии, этот элемент позволяет ассоциировать текущие наименования с исходными версиями.

Использование цифровых подписей обеспечивает целостность элементов, которые остаются неизменными после создания тега. Для того чтобы принять решение о подписывании конкретных элементов в теге, необходимо учитывать затраты на обеспечение дополнительной защиты в окружениях реализации и тестирования.

B.4 Руководитель разработки

Руководитель разработки работает непосредственно с менеджером по продукту, обеспечивая и четкое определение и полноту спецификаций продукта. Руководитель разработки работает с информацией, уровень детализации которой аналогичен уровню детализации информации, с которой работает менеджер по продукту.

Тем не менее, на этапе проектирования и реализации менеджер по продукту работает с более детализированной информацией. Ниже приводится сводный список сведений о реализации, которые необходимо принимать во внимание и по которым необходимо принимать решения перед переходом на этап реализации:

a) создание тега идентификации программного обеспечения: когда и как будет создаваться тег идентификации программного обеспечения?

Здесь необходимо учитывать несколько вариантов. Наилучший результат достигается, если учитывать конкретные обстоятельства каждого создателя программного обеспечения, циклы разработки и эксплуатационные и производственные процессы. Возможные варианты:

1) предварительно сформированный файл(ы) тега – включается в установочный диск, выбирается и копируется на целевую машину во время установки,

2) файл тега формируется в процессе установки – тег идентификации программного обеспечения создается в рамках процесса установки. Это позволяет включить в тег специальные установочные параметры, например, номер фактически устанавливаемой версии,

3) предварительно частично сформированный и обновляемый «на лету» файл тега – этот вариант позволяет выдавать тег идентификации программного обеспечения при первоначальной установке программного обеспечения и обновлять его при необходимости. Обновления могут заключаться в изменении состояния активации или других элементов по мере запуска и/или регистрации продукта;

b) файл тега идентификации программного обеспечения для объектов лицензирования множественных продуктов (например, наборов) – каким образом осуществляется создание и управление файлом тега?

Руководители разработки должны учитывать, где и как осуществляется создание и управление файлом(файлами) тега для компонентов набора. К вариантам создания тегов идентификации программного обеспечения могут относиться предоставление тега идентификации программного обеспечения для набора и отдельных тегов идентификации программного обеспечения для каждого отдельного продукта или создание одного тега идентификации программного обеспечения, описывающего весь набор. Кроме того, если продукт осуществляет валидацию своего тега, определяется, будет ли он выполнять валидацию отдельного приложения и набора или всех приложений, являющихся частью набора;

c) управление жизненным циклом файлов тега идентификации программного обеспечения – как меняется файл тега идентификации программного обеспечения вместе с изменениями установленного программного обеспечения?

Руководители разработки могут работать с несколькими сценариями жизненного цикла, воздействующими на теги идентификации программного обеспечения; такими сценариями могут быть следующие:

1) патчи и обновления – если патч ли обновление меняют версию продукта, файл тега должен отражать последнюю версию. Если элемент product version (версия продукта) был первоначально подписан, то подпись также должна быть обновлена,

2) отсутствующий файл тега идентификации программного обеспечения – файлы тега могут быть случайно стерты конечными пользователями или повреждены; в этих случаях программное обеспечение должно запустить механизмы самовосстановления для регенерации тега,

3) лицензии на пробное использование – после истечения срока действия лицензий на пробное использование обновления тега должны отражать, что период пробного использования завершен (т. е. должен быть обновлен элемент состояния активации (activation status)).

4) свидетельство взлома – если файл включает в себя подписанные элементы тега, программное обеспечение должно выполнять периодические проверки подписей с целью обнаружения возможного несанкционированного взлома тега. При обнаружении взлома тега программное обеспечение должно запустить механизм самовосстановления для регенерации тега. Руководитель разработки и менеджер по продукту должны совместно определить бизнес-политики в отношении взлома;

d) дополнительные соображения по реализации

1) как можно проверить тег на правильность во время цикла QA?,

2) наличие централизованного подразделения и реализация в привязке к конкретному продукту: создателям программного обеспечения среднего и крупного размеров настоятельно рекомендуется избавить группы по разработке продуктов от работ по реализации тегов и возложить эту функцию на централизованное подразделение по работе с тегами идентификации программного обеспечения,

3) при выпуске готового к использованию коммерческого программного продукта (COTS) поставщику программного обеспечения предоставляется специальная информация об «отпечатке» пакета (см. элемент package footprint, 8.4.10). Такая информация об «отпечатке» может быть размещена на сайте создателя программного обеспечения и использоваться для целей управления. Предоставление информации об «отпечатке» пакета значительно облегчит работу инструментария SAM и специалистов-практиков по использованию процессов SAM, которые смогут осуществлять автоматическую фильтрацию «известных» файлов, записей реестра и записей WMI, а также других составляющих, найденных во время обнаружения, и практиковать применение процессов SAM на базе исключений,

4) в случае использования «отпечатка» пакета обычно приводится ссылка на идентификатор URI – это место в большинстве случаев находится на домене создателя программного обеспечения. При предоставлении информации об «отпечатке» пакета руководитель разработки должен применять политику, обеспечивающую «свежесть» всей файловой информации, например, о патчах и незначительных релизах, регулярно выпускаемых на рынок. Элемент package footprint («отпечаток» пакета) разрешает указывать несколько записей в каждом файле, поэтому при ссылке на конкретный URI в один «отпечаток» пакета могут быть включены теги по нескольким версиям продукта.

Группа разработчиков, занимающаяся разработкой и управлением программным обеспечением, должна обеспечить интеграцию тега идентификации программного обеспечения в процессы разработки и управления жизненным циклом.

Приложение С
(справочное)

Сценарии действий и инструкции для поставщиков инструментария

С.1 Поставщики инструментария для обнаружения тегов

С.1.1 Введение

Инструментарий для обнаружения тегов должен уметь считывать данные из существующих тегов идентификации программного обеспечения. В организации потребителя программного обеспечения использовать инструментарий для обнаружения тегов может менеджер по аудиту, в задачу которого входит выверка прав на использование программного обеспечения с тегами идентификации программного обеспечения, или владелец процессов SAM, в задачу которого входит сбор и анализ информации.

Сценарии использования такого инструментария сторонних поставщиков могут быть основными и вспомогательными.

С.1.2 Основные сценарии использования

Инструментарий для обнаружения тегов должен обеспечивать следующую функциональность:

a) Обеспечивать целостность и единообразие данных, содержащихся в теге идентификации программного обеспечения.

Примечание – Если создатель тега (элемент tag_creator), создавший тег идентификации программного обеспечения, использует несколько элементов «software licensor identity» (идентификационные данные лицензиара программного обеспечения), «product identifier» (идентификатор продукта), «serial number» (серийный номер) или «stock keeping unit» (единица складского хранения), инструментарий для обнаружения тегов должен иметь возможность обращения к таблице распознавания программного обеспечения (выдаваемой создателем тега) для упрощения процессов выверки различных значений (см. 8.3.5, 8.4.14, 8.4.20, 8.4.21). Эта задача может потребовать включения расширенных элементов, предоставляемых создателем тега (элемент tag_creator).

Пример – Если программная компания А была приобретена программной компанией В, то компания В должна предоставить информацию о выверке для ранее выпущенного и снабженного тегами компанией А программного обеспечения. Или, если только позже компания А выпустит программный пакет (его новую версию с тем же или измененным элементом «software creator name» (имя создателя программного обеспечения), то компания А должна в элементах product_id и software_creator alias указать информацию, позволяющую автоматически идентифицировать отношения между двумя продуктами.

b) Выверять данные тегов идентификации программного обеспечения с данными соответствующих прав на использование программного обеспечения.

Примечание – Такая выверка может выполняться агентом на платформе или на административной консоли, собирающей данные от нескольких агентов. Для определения лицензионного соответствия программного обеспечения процессам выверки нужны данные как тега идентификации программного обеспечения, так и данные прав на использование программного обеспечения.

Пример – Инструментарий должен уметь идентифицировать различия между установками различных, но связанных между собой продуктов (по зависимости, по совокупности, по платформе или по версии продукта), например, различия между Microsoft® Excel®, Microsoft® Excel® Viewer (несопровождаемый), Microsoft® Office Standard 2000, Microsoft® Office Standard 2003 и Microsoft® Office XP Professional.

c) Считывать в процессе обнаружения все обязательные элементы (в том числе, если имеются, дополнительные элементы). При этом должны использоваться стандартные правила определения местоположения файла тега идентификации программного обеспечения и формат данных этого файла.

С.1.3 Вспомогательные сценарии использования

Инструментарий для обнаружения тегов должен обеспечивать следующую функциональность:

a) Использовать дополнительные элементы для упрощения процессов отслеживания использования. В этих случаях особую важность имеет элемент «usage identifier» (идентификатор использования) (см. 8.4.29).

b) Использовать дополнительные элементы (если имеются) для определения наличия программного ключа для авторизации использования продукта.

С.2 Поставщики инструментария для распространения программного обеспечения

С.2.1 Общие положения

Поставщики инструментария для распространения программного обеспечения создают инструментарий для упаковки, тестирования, распространения и установки программного обеспечения. С инструментарием для распространения программного обеспечения работают следующие конечные пользователи: администратор релизов, специалист практик по упаковке, отвечающий за упаковку программного обеспечения для развертывания, и специалист-практик по использованию процессов SAM, в задачу которого входит собственно развертывание.

Сценарии использования инструментария для распространения программного обеспечения могут быть основными и вспомогательными.

С.2.2 Добавление информации к тегам идентификации программного обеспечения для распространения: основные сценарии использования

Инструментарий для распространения программного обеспечения должен обеспечивать следующую функциональность:

a) создавать тег идентификации программного обеспечения «с нуля» в следующих типовых сценариях:

1) организация потребителя программного обеспечения разрабатывает программное обеспечение, например, простой файл скрипта или документ , содержащий макросы, которое необходимо квалифицировать как программный пакет для отслеживания,

2) организация потребителя программного обеспечения развертывает пакеты устаревшего программного обеспечения, не содержащие тегов идентификации программного обеспечения;

b) определять, должен или не должен быть доступен для изменения или удаления конкретный элемент;

c) разрешать переупаковку программного обеспечения посредством включения элементов, относящихся как к упаковке исходного программного обеспечения, так и к переупаковке.

Примечание – Такая переупаковка может включать в себя объединение нескольких программных пакетов в один программный пакет для распространения и развертывания;

Пример – Создание программного пакета внутри программного пакета. Создатель программного обеспечения создает тег идентификации программного обеспечения для продукта, упаковывая продукт в формат распространения. Затем для программного набора, содержащего исходный продукт, создается отдельный тег идентификации программного обеспечения. В процессе работы инструментария для распространения программного обеспечения в организации потребителя программного обеспечения может быть создан новый пакет для распространения программного обеспечения с собственным тегом идентификации программного обеспечения. После установки окончательного составного программного пакета все три тега идентификации программного обеспечения должны быть доступны для работы с ними.

С.2.3 Добавление информации к тегам идентификации программного обеспечения для распространения: вспомогательные сценарии использования

Инструментарий для распространения программного обеспечения должен уметь использовать доступные обязательные и дополнительные элементы для отслеживания и идентификации этапов управления релизами. К таким этапам относятся проектирование, создание, тестирование, утверждение, распространение и установка данного программного пакета.

С.2.4 Использование данных, содержащихся в тегах идентификации программного обеспечения для распространения: основные сценарии использования

Инструментарий для распространения программного обеспечения должен обеспечивать следующую функциональность:

a) обеспечивать целостность и единообразие данных, содержащихся в теге идентификации программного обеспечения;

b) Использовать теги идентификации программного обеспечения для автоматизации процессов упаковки.

С.2.5 Использование данных, содержащихся в тегах идентификации программного обеспечения для распространения: вспомогательные сценарии использования

Инструментарий для распространения программного обеспечения должен обеспечивать следующую функциональность:

a) определять по данным тега идентификации программного обеспечения, является ли конкретный элемент конфигурации программного обеспечения обновлением существующего тега или заменой другого тега.

Пример – Если один создатель программного обеспечения разрешает одновременную установку нескольких копий одного и того же продукта, инструмент распространения должен иметь возможность определения этого атрибута при установке. Если создатель программного обеспечения требует до установки нового программного обеспечения выполнить обновление или удалить предыдущую его версию, инструмент распространения также должен иметь возможность определения этого атрибута;

b) использовать дополнительные элементы «component association» (ассоциация компонентов) и «dependency» (зависимость) для определения зависимых элементов конфигурации программного обеспечения, которые должны быть установлены вместе с данной частью программного обеспечения (см. 8.4.2, 8.4.5);

c) использовать дополнительные элементы «digital signature» (цифровая подпись) и «data source» (источник данных) для определения того, должен ли программный пакет устанавливаться с учетом требований к безопасности установки (см. 6.1.11, 8.4.4).

Примечание – В том случае, если организация потребует, чтобы все элементы конфигурации программного обеспечения имели перед установкой цифровую подпись, то инструмент для распространения должен иметь возможность считывать из дополнительных элементов соответствующие цифровые подписи и проверять их правильность.

Приложение D
(справочное)

Сценарии действий и инструкции для потребителей программного обеспечения

D.1 Требования к специалисту-практику по использованию процессов SAM

Программы SAM успешно используются специалистами-практиками по использованию процессов SAM во многих областях, при этом главным преимуществом, обеспечивающим удовлетворение разнообразных требований IT-окружения, является способность выполнять точную инвентаризацию программного обеспечения. Такими требованиями могут быть следующие:

a) управление лицензиями на программное обеспечение – сюда относятся задачи обеспечения лицензионного соответствия программного обеспечения, принятия решений, связанных с финансовыми аспектами, и управления жизненным циклом активов:

1) действия по обеспечению лицензионного соответствия программного обеспечения определяют, выполняет ли организация условия соглашений о предоставлении лицензий на программное обеспечение. Определение базируется на анализе прав на использование программного обеспечения и анализе разрешенных количеств в сравнении с фактическими атрибутами использования и установленными количествами,

2) к финансовым решениям, принимаемым в отношении лицензий на программное обеспечение, могут относиться следующие: приобретение дополнительных или новых лицензий на программное обеспечение, обновление существующих лицензий на программное обеспечение, обновление средств технического обслуживания и поддержки, действия по слиянию/приобретению/отделению, риски аудиторских проверок, решения по непредвиденным лицензионным расходам и прочие решения,

3) к действиям по управлению жизненным циклом, относящимся к лицензиям на программное обеспечение, относятся процессы получения, развертывания, восстановления действительности лицензий (по мере возникновения таких требования у конечного пользователя) и восстановления действительности лицензий по мере использования физических активов;

b) стабильность платформы – возможность выявления проблем совместимости между программными пакетами и операционными системами, а также обеспечение совместимости файлов данных, обеспечивающих работу конечных пользователей;

c) IT-безопасность – сюда входят все аспекты безопасности (защита от вирусов, защита от вредоносного программного обеспечения, ограничение использования несанкционированного программного обеспечения, требования к шифрованию и пр.) и эффективная идентификация подверженных риску областей организации.

В эффективной реализации процессов SAM участвуют множество исполнителей, и эти исполнители могут в различных организациях называться по-разному. Соответственно, в настоящем приложении упор делается не на описании конкретных должностных функций, а на определении ряда высокоуровневых сценариев для каждой из перечисленных выше областей.

D.2 Управление лицензиями на программное обеспечение

D.2.1 Общие положения

Управление лицензиями на программное обеспечение является частью общей программы SAM и используется для обеспечения того, чтобы программные активы приобретались и использовались способом, соответствующим бизнес-целям организации. Существует множество сценариев определения того, как именно должны использоваться теги идентификации программного обеспечения в рамках программы управления лицензиями на программное обеспечение, однако в настоящем приложении рассматриваются только три основных области, в которых использование тегов может представлять особую важность.

D.2.2 Действия по обеспечению соответствия

Большая часть программного обеспечения лицензируется, а не продается. Другими словами, программное обеспечение поставляется с определенными правами на использование, предоставляемыми покупателю программного обеспечения. Практически во всем лицензируемом программном обеспечении данными, требуемыми для проверки соответствия, являются данные о точном количестве установленных копий наименований программного обеспечения.

Основные сценарии определения соответствия тегов идентификации программного обеспечения предполагают обнаружение и постоянную инвентаризацию названий программного обеспечения, установленного на всех вычислительных устройствах, находящихся в распоряжении организации. Для выполнения этих сценариев специалист-практик по использованию процессов SAM может использовать сторонний инструментарий или же мо-

жет использовать собственные инструменты инвентаризации, при условии, что такие инструменты могут собирать все теги идентификации программного обеспечения со всех устройств, находящихся под управлением организации. К информации, требуемой для выполнения данного процесса, относится следующая (ниже приводится минимальный список, взятый из стандарта ИСО/МЭК 19770-1:2006):

a) устройство, на котором обнаружена инвентарная позиция;

b) название программного обеспечения;

c) владелец программного обеспечения;

d) состояние (тестируется/в производстве/оценивается/лицензировано);

e) Версия программного обеспечения.

Приведенный выше список является абсолютно минимальным, и ряд инструментов SAM могут требовать разумной детализации указанных выше позиций. Для того чтобы получить и поддерживать надлежащее лицензионное соответствие, в дополнительной информации, выдаваемой тегами идентификации программного обеспечения, должны содержаться, в частности, следующие составляющие (список может быть продолжен):

a) данные о том, нужны ли права на использование;

b) имя создателя программного обеспечения;

c) имена любых других предыдущих создателей программного обеспечения или имена продуктов, относящихся к данному названию;

d) данные о типе программного обеспечения;

e) является ли обнаруженный программный пакет частью набора;

f) серийный номер обнаруженного названия программного обеспечения;

g) SKU (складская единица хранения) обнаруженного названия программного обеспечения.

Для точного определения программного пакета и его структуры служат многие данные, в том числе информация о том, как именно программный пакет был установлен на вычислительное устройство (т. е. было ли программное обеспечение установлено с CD или с использованием средств управления настольными системами). Традиционный инструментарий SAM выдает минимальную информацию, которая пополняется по мере сбора данных, содержащихся в тегах идентификации программного обеспечения.

Предполагается, что процесс обнаружения будет собирать все теги идентификации программного обеспечения, обнаруженные на устройстве, вместе с соответствующими данными о конкретном вычислительном устройстве (эти данные не включаются в тег идентификации программного обеспечения), такими как:

a) имя машины;

b) IP-адрес машины;

c) MAC-адрес(а) машины;

d) Операционная система, установленная на вычислительном устройстве.

Сбор информации о другом оборудовании будет осуществляться в ходе процесса инвентаризации. Это информация будет необходима для оформления отчета о соответствии. Предполагается, что после определения стандартов электронных прав на использование для оформления отчета о соответствии будет использоваться информация, собранная в правах на использование.

Для составления надлежащих отчетов о соответствии организации должны обеспечить сбор и хранение всех тегов идентификации программного обеспечения вместе с информацией об оборудовании.

Процесс проверки соответствия требует проведения начальной базовой инвентаризации (первоначальных процессов обнаружения и инвентаризации), а также постоянного выполнения процессов инвентаризации. Этот процесс позволяет специалисту-практику по использованию процессов SAM идентифицировать различия на уровнях установки и соответствия. Соответственно, специалисты-практики по использованию процессов SAM, использующие инструментарий, должны в течение некоторого времени сохранять данные предыдущих инвентаризаций, как это определено организационной политикой.

D.2.3 Принятие решений, связанных с финансовыми аспектами

При принятии решений, связанных с финансовыми аспектами, требуется информация, аналогичная информации об идентификации и инвентаризации программного обеспечения, используемой для целей проверки соответствия, однако эти данные используются несколько по-другому.

Задача основного сценария финансовых действий состоит в отслеживании программных активов, находящихся в собственности организации, и обеспечении их эффективного использования в соответствии с финансовыми целями организации. Финансовым аналитикам данные инвентаризации программного обеспечения могут потребоваться для того, чтобы выяснить, как именно программное обеспечение отвечает этим целям. Этот процесс требует наличия полной и точной информации об инвентаризации, а также ссылок на другую внешнюю информацию, например:

a) заказы на поставку программного обеспечения;

b) расценки на программное обеспечение;

c) расценки на техническое обслуживание;

d) расценки на обновления;

e) стоимости замещения программных активов на момент слияний/приобретений/отделений.

Финансовым аналитикам может потребоваться дополнительная информация, содержащаяся непосредственно в тегах идентификации программного обеспечения. Например, финансовому аналитику, обеспокоенному риском несоответствия лицензий на программное обеспечение, для оценки финансового риска, которому может быть подвержена организация, возможно, потребуется знать расценки и условия предоставления любых приобретенных лицензий. Кроме того, финансовый аналитик, возможно, захочет ознакомиться с данными по использованию, чтобы оценить, как используется установленное программное обеспечение, или возможны ли переговоры о заключении соглашения об обслуживании, связанного с меньшими эксплуатационными расходами. Тег идентификации программного обеспечения содержит данные о том, что именно должно отслеживаться для определения степени использования программного обеспечения.

Примечание – Вопросы выпуска лицензионных прав будут рассмотрены в будущей части стандарта ИСО/МЭК 19770.

В теге идентификации программного обеспечения содержится другая полезная информация для финансового аналитика, которая может быть использована непосредственно для анализа эффективности использования программных активов. Для успешной работы аналитик должен использовать те же точные и актуальные данные инвентаризации, которые используют в своей работе группы управления соответствием или управления жизненным циклом.

D.2.4 Действия по управлению жизненным циклом

К программному обеспечению, как и к аппаратному обеспечению, применяется процесс управления жизненным циклом. Для управления жизненным циклом специалист-практик по использованию процессов SAM должен применять особые процессы. Для выполнения этих процессов могут пригодиться данные, содержащиеся в теге идентификации программного обеспечения.

К основным сценариям использования, применяемым при управлении жизненным циклом, относятся получение приобретенного программного обеспечения, развертывание и восстановление действия лицензий и отслеживание лицензий.

a) Получение программного обеспечения – организации могут получать установочные носители либо физически (на одного или несколько пользователей), либо посредством электронной загрузки данных для установки приложения. Чтобы правильно ассоциировать полученное программное обеспечение с существующими инвентарными позициями, получающий департамент должен собрать и сохранить сведения, содержащиеся в заказах на покупку, лицензионные сертификаты и другую информацию, ассоциирующую полученные записи с программным приложением и информацией, содержащейся в тегах идентификации программного обеспечения.

b) Развертывание и восстановление действия лицензий – работники организаций могут часто менять свои должности или оборудование, на котором они работают. При управлении жизненным циклом программного обеспечения важно знать, каким именно программным обеспечением разрешено пользоваться каждому работнику и каждой должности в организации. Зная точные и актуальные данные инвентаризации, лицензии можно удалять и переназначать в рамках организации.

c) Отслеживание установок программного обеспечения – организации часто создают процедуры установки, привязывающие программное обеспечение к окружению, в котором оно будет работать. Работникам, занятым выполнением процессов управления жизненным циклом, могут потребоваться в теге идентификации программного обеспечения дополнительные элементы, с помощью которых они смогут указывать используемый процесс выпуска релизов для тестирования и развертывания программного обеспечения на вычислительных устройствах конечных пользователей (таким элементами могут быть, например, «release verification» (верификация релиза) и «release rollout» (развертывание релиза). Кроме того, если требуется обеспечение постоянного контроля инвентаризации, такие дополнительные элементы можно использовать для определения того, было ли программное обеспечение установлено с использованием разрешенных в организации методов, или было установлено без разрешения, например, в форме копии программного обеспечения, приобретенной в розницу.

D.3 Стабильность платформы

Стабильность платформы затрагивает целый ряд аспектов, включая возможность тестирования нового программного и аппаратного обеспечения в известном операционном окружении с целью обеспечения совместимости, а также для того, чтобы конечные пользователи знали о том, что создаваемые ими файлы данных могут использоваться другими сотрудниками в пределах организации.

К основным сценариям использования данного процесса относятся следующие:

a) валидация согласованного общего системного окружения – этот процесс требует проведения начальной базовой инвентаризации, а также регулярных плановых инвентаризаций. Данные, полученные в результате начальной базовой инвентаризации и данные последней инвентаризации сравниваются между собой с целью проверки того, что все приложения в общем операционном окружении имеют правильные номера версий и патчей, и что отдельные сотрудники не обновляют программное обеспечение или не устанавливают различные программные инструменты, не совместимые с общим операционным окружением;

b) валидация нового программного и аппаратного обеспечения – этот процесс требует определения и поддержания в текущем состоянии общего операционного окружения (см. выше). По мере поступления в организацию нового программного и аппаратного обеспечения специалисты по контролю качества должны убеждаться в том, что новые элементы не вступают в конфликт ни с какими другими элементами в определенном общем операционном окружении;

c) обработка обновлений – время от времени на программное обеспечение могут устанавливаться обновления. В этих случаях должна осуществляться проверка того, что более новая версия программного обеспечения корректно работает с существующими названиями программного обеспечения и, если необходимо, проверка того, что файлы данных, созданные кем-либо с помощью нового программного обеспечения, могут использоваться более старыми версиями программного обеспечения. На основании информации, обнаруженной при тестировании в общем операционном окружении, составляются и выполняются планы обновлений, обеспечивающие минимальные неудобства для организации при необходимости обновлений программного обеспечения.

D.4 IT-безопасность

Организации отвечают за множество аспектов IT-безопасности. Компании устанавливают на свои вычислительные устройства межсетевые экраны, ограничивающие доступ в Интернет, антивирусные программы и сканеры программ-шпионов, а некоторые из этих организаций отслеживают сетевую активность. Несанкционированное программное обеспечение, устанавливаемое пользователями на свои компьютеры, сопряжено с высоким (часто недооцениваемым) риском безопасности.

К основным сценариям использования данного процесса относятся следующие:

a) идентификация несанкционированного программного обеспечения – этот процесс позволяет специалисту-практику по использованию процессов SAM использовать данные инвентаризации программного обеспечения и идентифицировать любые установленные программные пакеты, не входящие в список программного обеспечения, утвержденный организацией. При выполнении этого процесса могут использоваться данные тега идентификации программного обеспечения, выдающие «отпечаток», идентифицирующий файлы, относящиеся к конкретному программному пакету. Служба IT-безопасности сможет быстро отфильтровать все известные и утвержденные файлы и оперативно проверить все другие обнаруженные исполняемые файлы. Служба IT-безопасности может создать собственный тег идентификации программного обеспечения, включающий данные об известных и разрешенных организацией к использованию файлах, и использовать этот тег для последующей фильтрации списка. Таким образом, процесс фильтрации может идентифицировать неизвестные исполняемые файлы, проверять, разрешены ли они к использованию в пределах организации и либо принять, либо отклонить программное обеспечение в соответствии с организационной политикой;

b) проверка соответствующих версий и файлов данных установленных приложений безопасности, например, сканеров программ-шпионов и антивирусного программного обеспечения. Использование актуальных и точных данных инвентаризации, аналогичных тем, которые используются в других группах организации, позволит запускать этот процесс в автоматизированном режиме.

Приложение Е
(справочное)

Теги идентификации программного обеспечения для элементов, не являющихся программным обеспечением

Е.1 Лицензируемые продукты

Е.1.1 Введение

Программное обеспечение – единственное из множества лицензируемых элементов, которое организации потребителя программного обеспечения может понадобиться отслеживать с целью выверки обнаруженных или используемых элементов с правами на использование. Другими примерами лицензируемых элементов могут служить аналитические отчеты, шрифты, графические объекты, словари и пр.

Теги идентификации программного обеспечения могут использоваться для идентификации любых типов данных. Поскольку названия программного обеспечения являются лицензируемыми данными, и это сопряжено со значительными торговыми расходами, в данной части настоящего стандарта особое внимание уделяется проблеме идентификации программного обеспечения. Несмотря на то, что данные, содержащиеся в теге идентификации программного обеспечения, в основном ориентированы на программные продукты, теги идентификации программного обеспечения могут также использоваться для отслеживания других лицензируемых элементов.

В данной части настоящего стандарта не приводятся точные указания насчет того, каким образом различные продукты, не являющиеся программными продуктами, могут использовать теги идентификации программного обеспечения, поскольку для таких различных элементов могут быть определены не сопоставимые между собой процедуры установки, или к ним могут применяться альтернативные сценарии использования.

В тех случаях, когда данные являются лицензируемыми, а инфраструктура для управления этими данными не определена, могут использоваться различные подходы. К таким случаям, в частности, могут относится следующие.

Е.1.2 Информация встроенного тега

В многие файлы разрешается вставлять дополнительные данные без ущерба для лицензируемых данных, уже содержащихся в файле. Например, в PDF-файлы, обычно используемые при составлении аналитических отчетов, а также в другие материалы, защищенные авторским правом, можно встраивать файлы вложений. Встраивая в PDF-файл тег, определенный в соответствии с данной частью настоящего стандарта, специалисты по лицензированию, отслеживающие права на использование, могут оперативно находить сведения о документе.

Е.1.3 Хранение тегов идентификации в том же каталоге, в котором хранятся лицензированные продукты

Часто элементы, связанные с правами на использование, представляют собой «копии, установленные» в определенный каталог. Другими словами, необходимая информация копируется в каталог и при необходимости берется из этого каталога. В этих случаях тег идентификации программного обеспечения может содержаться в исходных файлах, копируемых на вычислительное устройство. Инструментарий для обнаружения тегов должен проверять правильность всех тегов идентификации программного обеспечения, хранящихся в системе в каталогах, отличных от каталогов по умолчанию для конкретной платформы. В этих случаях инструментарий SAM также должен идентифицировать отсутствие копии в главном системном каталоге. Такой элемент будет помечен как элемент для последующей квалификации в качестве программного обеспечения, которое было установлено на съемный носитель и затем перемещено, или как другой тип данных, к которому также могут применяться права на использование.

Е.2 Использование расширенных элементов

Для файлов данных, использующих данные тегов идентификации программного обеспечения, рекомендуется предоставлять дополнительную информацию в виде расширенных элементов. Эти дополнительные данные могут пригодиться специалистам-практикам по использованию процессов SAM для идентификации типа данных, на которые ссылается тег, а также для определения организационного подразделения внутри компании, отвечающего за управление данными этого типа.

Приложение F
(справочное)

Авторское право и теги идентификации программного обеспечения

F.1 Сведения от авторском праве

Авторское право в отношении данной части настоящего стандарта принадлежит ИСО/МЭК. Использование и реализация настоящего документа не связаны ни с какими выплатами авторского вознаграждения, за исключением оплаты стоимости изготовления копий самого стандарта.

Данная часть настоящего стандарта была переработана с учетом следующих предположений относительно данных авторского права, содержащихся в теге(тегах) идентификации программного обеспечения, созданных в соответствии с этим стандартом:

a) предполагается, что данные об авторском праве, содержащиеся в тегах, созданных создателями программного обеспечения, будут такими же, что и для самого программного обеспечения. Ограничения на использование тегов не накладываются, кроме тех случаев, когда тег должен использоваться исключительно в целях управления конкретным соответствующим программным обеспечением;

b) рекомендуется, чтобы создатели программного обеспечения разрешали последующее распространение созданных ими и защищенных их авторским правом тегов идентификации программного обеспечения, при условии, что информация, содержащаяся в тегах не будет меняться. Предоставление данного права позволит поставщикам инструментария SAM и специалистам-практикам по использованию процессов SAM обращаться к информации, содержащейся в теге идентификации программного обеспечения, которая может оказаться полезной для идентификации программного обеспечения в определенных ситуациях (например, в случаях, когда тег идентификации программного обеспечения был непреднамеренно удален с вычислительного устройства);

c) предполагается, что, если в тегах, созданных физическими лицами или организациями, не являющимися создателями программного обеспечения (например, поставщиками инструментария для использования с устаревшим программным обеспечением), содержатся данные об авторском праве, то это обстоятельство будет налагать дополнительные ограничения, заключающиеся в том, что данные тега могут использоваться только в программных инструментах данного поставщика инструментария (или других указанных поставщиков инструментария);

d) предполагается, что никакие авторские права, связанные с изменениями существующих тегов или добавлениями к существующим тегам, не действуют. При необходимости указания отдельных авторских прав должен быть создан родительский тег, в котором будет указана дополнительная или измененная информация с указанием соответствующих авторских прав.

Приложение G
(справочное)

Определение схемы XML (XSD)

G.1 Введение

В следующем XSD приводится определение тега идентификации программного обеспечения. Файл XSD должен быть доступен по следующему адресу:

http://standards.iso.org/iso/19770/-2/2009/schema.xsd

G.2 Схема XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" targetNamespace="gost.e-ecolog.ru
schema.xsd"
xmlns:swid="gost.e-ecolog.ru so/19770/-2/2008/schema.xsd"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:import namespace="gost.e-ecolog.ru schemaLocation="gost.e-ecolog.ru
gost.e-ecolog.ru " />
<xs:annotation>
<xs:documentation>
Schema for ISO-IEC 19770-2 Software Identification Tags gost.e-ecolog.ru
Revision: 1,0
Copyright 2009 ISO/IEC, all rights reserved
This XML Scheme Document (XSD) may be accessed, stored, copied and transferred without authorization from
ISO or its members on the condition that it is not modified and that there is no charge associated with access to this XSD
file. Copyright remains with ISO.
</xs:documentation>
</xs:annotation>
<!-- Root tag -->
<xs:element name="software_identification_tag" type="swid:SoftwareIdentificationTagComplexType" />
<!-- Software Asset Tag structure definition -->
<xs:complexType name="SoftwareIdentificationTagComplexType">
<xs:sequence>
<!-- Mandatory Elements -->
<xs:element name="entitlement_required_indicator" type="swid:Boolean" />
<xs:element name="product_title" type="swid:Token" />
<xs:element name="product_version" type="swid:ProductVersionComplexType" />
<xs:element name="software_creator" type="swid:EntityComplexType" />
<xs:element name="software_licensor" type="swid:EntityComplexType" />
<xs:element name="software_id" type="swid:SoftwareIdComplexType" />
<xs:element name="tag_creator" type="swid:EntityComplexType" />
<!-- Optional Elements -->
<xs:element minOccurs="0" maxOccurs="unbounded" name="abstract" type="swid:AbstractComplexType" />
<xs:element minOccurs="0" name="component_of" type="swid:ListOfSoftwareIdsComplexType" />
<xs:element minOccurs="0" name="complex_of" type="swid:ListOfSoftwareIdsComplexType" />
<xs:element minOccurs="0" name="data_source" type="swid:Token" />
<xs:element minOccurs="0" name="dependency" type="swid:ListOfSoftwareIdsComplexType" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="elements_owner" type="swid:ElementsOwnerComp
lexType" />
<xs:element minOccurs="0" name="installation_details" type="swid:InstallationDetailsComplexType" />
<xs:element minOccurs="0" name="keywords" type="swid:KeywordsComplexType" />
<xs:element minOccurs="0" name="license_linkage" type="swid:LicenseLinkageComplexType" />
<xs:element minOccurs="0" name="package_footprint" type="swid:PackageFootprintComplexType" />
<xs:element minOccurs="0" name="packager" type="swid:PackagerComplex-Type" />
<xs:element minOccurs="0" name="product_category" type="swid:CategoryComplexType" />
<xs:element minOccurs="0" name="product_family" type="swid:Token" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="product_id" type="swid:Token" />
<xs:element minOccurs="0" name="release_date" type="swid:DateTime" />
```

```
        <xs:element minOccurs="0" name="release_id" type="swid:Token" />
        <xs:element minOccurs="0" name="release_package" type="swid:ReleaseComplexType" />
        <xs:element minOccurs="0" name="release_rollout" type="swid:ReleaseComplexType" />
        <xs:element minOccurs="0" name="release_verification" type="swid:ReleaseComplexType" />
        <xs:element minOccurs="0" name="serial_number" type="swid:Token" />
        <xs:element minOccurs="0" name="sku" type="swid:Token" />
        <xs:element minOccurs="0" name="software_creator_alias" type="swid:EntityDataComplexType" />
        <xs:element minOccurs="0" name="software_licensor_alias" type="swid:EntityDataComplexType" />
        <xs:element minOccurs="0" name="supported_languages" type="swid:SupportedLanguagesComplexType" />
        <xs:element minOccurs="0" name="tag_creator_alias" type="swid:EntityDataComplexType" />
        <xs:element minOccurs="0" name="tag_creator_copyright" type="swid:TagCreatorCopyrightComplexType" />
        <xs:element minOccurs="0" maxOccurs="unbounded" name="tag_version" type="swid:TagVersionComplexType"
/>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="upgrade_for" type="swid:UpgradeForComplexType
" />
        <xs:element minOccurs="0" maxOccurs="unbounded" name="usage_identifier" type="swid:UsageComplexType" />
        <xs:element minOccurs="0" name="validation" type="swid:ValidationComplexType" />
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="ds:Signature" />
        <!--Extended Information -->
        <xs:element minOccurs="0" maxOccurs="unbounded" name="extended_information" type="swid:ExtendedInforma
tionComplexType" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <!-- Mandatory Elements complex types -->
        <xs:complexType name="ProductVersionComplexType">
        <xs:sequence>
        <xs:element name="name" type="swid:Token" />
        <xs:element name="numeric" type="swid:NumericVersionComplexType" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="EntityComplexType">
        <xs:sequence>
        <xs:element name="name" type="swid:Token" />
        <xs:element name="regid" type="swid:RegistrationId" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="SoftwareIdComplexType">
        <xs:sequence>
        <xs:element name="unique_id" type="swid:Token" />
        <xs:element name="tag_creator_regid" type="swid:RegistrationId" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <!-- Optional Elements complex types -->
        <xs:complexType name="AbstractComplexType">
        <xs:simpleContent>
        <xs:extension base="swid:String">
        <xs:attribute use="optional" default="en" name="lang" type="xs:token" />
        </xs:extension>
        </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="ListOfSoftwareIdsComplexType">
        <xs:sequence>
        <xs:element maxOccurs="unbounded" name="software_id" type="swid:SoftwareIdComplexType" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="ElementsOwnerComplexType">
        <xs:sequence>
```

```
<xs:element minOccurs="0" name="owner_name" type="swid:Token" />
<xs:element name="owner_regid" type="swid:RegistrationId" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="element_id" type="swid:IDREF" />
</xs:sequence>
<xs:attributeGroup ref="swid:default" />
</xs:complexType>
<xs:complexType name="InstallationDetailsComplexType">
<xs:sequence>
<xs:element minOccurs="0" name="location_platform" type="swid:Token" />
<xs:element minOccurs="0" name="location_installation" type="swid:Token" />
<xs:element minOccurs="0" name="installation_instance" type="swid:Token" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="installation_locale" type="swid:Token" />
<xs:element minOccurs="0" name="installation_target_id" type="swid:Token" />
</xs:sequence>
<xs:attributeGroup ref="swid:default" />
</xs:complexType>
<xs:complexType name="KeywordsComplexType">
<xs:sequence>
<xs:element minOccurs="0" maxOccurs="unbounded" name="keyword" type="swid:Token" />
</xs:sequence>
<xs:attributeGroup ref="swid:default" />
</xs:complexType>
<xs:complexType name="LicenseLinkageComplexType">
<xs:sequence>
<xs:element minOccurs="0" maxOccurs="unbounded" name="activation_status" type="swid:Token" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="channel_type" type="swid:Token" />
<xs:element minOccurs="0" name="channel_name" type="swid:Token" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="customer_type" type="swid:Token" />
</xs:sequence>
<xs:attributeGroup ref="swid:default" />
</xs:complexType>
<xs:complexType name="PackageFootprintComplexType">
<xs:choice>
<xs:element name="external_description" type="swid:AnyURI" />
<xs:sequence>
<xs:element minOccurs="0" name="primary" type="swid:PackageFootprintModuleComplexType" />
<xs:element minOccurs="0" name="secondary" type="swid:PackageFootprintModuleComplexType" />
<xs:element minOccurs="0" name="related" type="swid:PackageFootprintModuleComplexType" />
</xs:sequence>
</xs:choice>
<xs:attributeGroup ref="swid:default" />
</xs:complexType>
<xs:complexType name="PackageFootprintModuleComplexType">
<xs:choice>
<xs:element name="referenced" type="swid:AnyURI" />
<xs:sequence>
<xs:element minOccurs="0" maxOccurs="unbounded" name="file" type="swid:
PackageFootprintFileComplexType" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="os_configuration_
record"
type="swid:PackageFootprintOsConfigurationRecordComplexType" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="other" type="swid:
PackageFootprintOtherComplexType" />
</xs:sequence>
</xs:choice>
<xs:attributeGroup ref="swid:default" />
</xs:complexType>
<xs:complexType name="PackageFootprintFileComplexType">
<xs:sequence>
<xs:element name="name" type="swid:Token" />
<xs:element minOccurs="0" maxOccurs="unbounded" name="size" type="swid:
UInt" />
```

```
        <xs:element minOccurs="0" maxOccurs="unbounded" name="md5" type="swid:
MD5" />
        <xs:element minOccurs="0" maxOccurs="unbounded" name="version" type="swid:
Token" />
        <xs:element minOccurs="0" maxOccurs="unbounded" name="other" type="swid:
PackageFootprintOtherParamComplexType" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="PackageFootprintOsConfigurationRecordComplexType»>
        <xs:sequence>
        <xs:element name=»type» type=»swid:Token» />
        <xs:element minOccurs="0" name="path" type="swid:Token" />
        <xs:element minOccurs="0" name="name" type="swid:Token" />
        <xs:element minOccurs="0" name="internal_path" type="swid:Token" />
        <xs:element maxOccurs="unbounded" name="entry" type="swid:ConfigEntryFootprintComplexType" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="ConfigEntryFootprintComplexType">
        <xs:sequence>
        <xs:element minOccurs="0" name="name" type="swid:Token" />
        <xs:element minOccurs="0" name="value" type="swid:Token" />
        <xs:element minOccurs="0" name="type" type="swid:Token" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="PackageFootprintOtherComplexType">
        <xs:sequence>
        <xs:element maxOccurs="unbounded" name="param" type="swid:PackageFootprintOtherParamComplexType" />
        </xs:sequence>
        <xs:attribute name="type" type="xs:token" use="required" />
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="PackageFootprintOtherParamComplexType">
        <xs:simpleContent>
        <xs:extension base="swid:Token">
        <xs:attribute name="name" type="xs:token" use="required" />
        </xs:extension>
        </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="PackagerComplexType">
        <xs:sequence>
        <xs:element name="by" type="swid:Token" />
        <xs:element name="part" type="swid:Token" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="CategoryComplexType">
        <xs:sequence>
        <xs:element name="UNSPSC_ver" type="swid:Token" />
        <xs:element name="segment_title" type="swid:Token" />
        <xs:element name="family_title" type="swid:Token" />
        <xs:element name="class_title" type="swid:Token" />
        <xs:element name="commodity_title" type="swid:Token" />
        <xs:element name="code" type="swid:UnspscIdType" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="ReleaseComplexType">
        <xs:sequence>
        <xs:element name="sign_off" type="swid:Token" />
```

```
        <xs:element name="sign_off_date" type="swid:DateTime" />
        <xs:element name="by" type="swid:Token" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="SupportedLanguagesComplexType">
        <xs:sequence>
        <xs:element maxOccurs="unbounded" name="language" type="swid:Token" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="EntityDataComplexType">
        <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="alias" type="swid:
AliasDetailsComplexType" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="AliasDetailsComplexType">
        <xs:sequence>
        <xs:element name="alias_name" type="swid:Token" />
        <xs:element name="alias_regid" type="swid:RegistrationId" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="TagCreatorCopyrightComplexType">
        <xs:simpleContent>
        <xs:extension base="swid:String">
        <xs:attribute use="optional" default="en" name="lang" type="xs:token" />
        </xs:extension>
        </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="TagVersionComplexType">
        <xs:sequence>
        <xs:element name="name" type="swid:Token" />
        <xs:element name="regid" type="swid:RegistrationId" />
        <xs:element name="numeric_version" type="swid:NumericVersionComplex
Type" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="UpgradeForComplexType">
        <xs:sequence>
        <xs:element maxOccurs="unbounded" name="upgrade_id" type="swid:SoftwareIdComplexType" />
        <xs:element minOccurs="0" name="upgrade_description" type="swid:String" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="UsageComplexType">
        <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="filename" type="swid:Token" />
        <xs:element minOccurs="0" maxOccurs="unbounded" name="processname" type="swid:UsageDetailComplexTy
pe" />
        <xs:element minOccurs="0" maxOccurs="unbounded" name="URI" type="swid:UsageDetailComplexType" />
        </xs:sequence>
        <xs:attributeGroup ref="swid:default" />
        </xs:complexType>
        <xs:complexType name="ValidationComplexType">
        <xs:sequence>
        <xs:element name="validation_call" type="swid:Token" />
        <xs:element minOccurs="0" name="last_validated_by" type="swid:Token" />
        <xs:element minOccurs="0" name="last_validated_date" type="swid:DateTime" />
```

```
<xs:element minOccurs="0" name="last_validated_result" type="swid:Token" />
</xs:sequence>
<xs:attributeGroup ref="swid:default" />
</xs:complexType>
<!-- Extended Information type definition -->
<xs:complexType name="ExtendedInformationComplexType">
<xs:sequence>
<xs:any minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
<xs:attributeGroup ref="swid:default" />
</xs:complexType>
<!-- Other complex types -->
<xs:complexType name="NumericVersionComplexType">
<xs:sequence>
<xs:element name="major" type="swid:UInt" />
<xs:element name="minor" type="swid:UInt" />
<xs:element name="build" type="swid:UInt" />
<xs:element name="review" type="swid:UInt" />
</xs:sequence>
<xs:attributeGroup ref="swid:default" />
</xs:complexType>
<xs:complexType name="UsageDetailComplexType">
<xs:simpleContent>
<xs:extension base="swid:Token">
<xs:attribute name="type" default="literal" use="optional">
<xs:simpleType>
<xs:restriction base="xs:token">
<xs:enumeration value="literal" />
<xs:enumeration value="regexp" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="GUIDType">
<xs:simpleContent>
<xs:restriction base="swid:Token">
<xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}" />
</xs:restriction>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="UnspscIdType">
<xs:simpleContent>
<xs:restriction base="swid:ULong">
<xs:pattern value="[0-9]{8}" />
</xs:restriction>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="MD5">
<xs:simpleContent>
<xs:restriction base="swid:HexBinary">
<xs:length value="16" />
</xs:restriction>
</xs:simpleContent>
</xs:complexType>
<!-- Complex types for simple types + attributes -->
<xs:complexType name="RegistrationId">
<xs:simpleContent>
<xs:restriction base="swid:Token">
<xs:pattern value="regid\.[0-9]{4}-((0[1-9])|(1[0-2]))\.[a-zA-Z]{2,63}(\.[a-zA-Z0-9]([a-zA-Z0-9\-]{0,61}[a-zA-
Z0-9])?)+(,.+)?" />
```

```
</xs:restriction>
</xs:simpleContent>
</xs:complexType>
<xs:attributeGroup name="default">
<xs:attribute name="id" type="xs:ID" use="optional" />
</xs:attributeGroup>
<xs:complexType name="String">
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attributeGroup ref="swid:default" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="Token">
<xs:simpleContent>
<xs:extension base="xs:token">
<xs:attributeGroup ref="swid:default" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="IDREF">
<xs:simpleContent>
<xs:extension base="xs:IDREF">
<xs:attributeGroup ref="swid:default" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="Boolean">
<xs:simpleContent>
<xs:extension base="xs:boolean">
<xs:attributeGroup ref="swid:default" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="UInt">
<xs:simpleContent>
<xs:extension base="xs:unsignedInt">
<xs:attributeGroup ref="swid:default" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="ULong">
<xs:simpleContent>
<xs:extension base="xs:unsignedLong">
<xs:attributeGroup ref="swid:default" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="HexBinary">
<xs:simpleContent>
<xs:extension base="xs:hexBinary">
<xs:attributeGroup ref="swid:default" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="DateTime">
<xs:simpleContent>
<xs:extension base="xs:dateTime">
<xs:attributeGroup ref="swid:default" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
```

```
<xs:complexType name="AnyURI">
<xs:simpleContent>
<xs:extension base="xs:anyURI">
<xs:attributeGroup ref="swid:default" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:schema>
```

Приложение Н
(справочное)

Расширенные примеры

Н.1 Пример 1

Следующий пример приводится только в информативных целях. Фактические значения, представленные в образце тега, являются фиктивными и не должны использоваться ни в каких производственных целях.

Пример 1: Adobe® Photoshop® CS (в том числе расширенная информация)

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:software_identification_tag
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:swid="http://standards.iso.org/iso/19770/-2/2008/schema.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="gost.e-ecolog.ru                                    software_identification_tag.xsd ">
<!-- Mandatory elements -->
<swid:entitlement_required_indicator id="e8_3_1">true</swid:entitlement_required_indicator>
<swid:product_title id="e8_3_2">Adobe Photoshop CS3</swid:product_title>
<swid:product_version id="e8_3_3">
<swid:name id="e8_3_3sub1">10.2</swid:name>
<swid:numeric id="e8_3_3sub2">
<swid:major id="e8_3_3sub2sub1">10</swid:major>
<swid:minor id="e8_3_3sub2sub2">2</swid:minor>
<swid:build id="e8_3_3sub2sub3">0</swid:build>
<swid:review id="e8_3_3sub2sub4">0</swid:review>
</swid:numeric>
</swid:product_version>
<swid:software_creator id="e8_3_4">
<swid:name>Adobe Systems Incorporated</swid:name>
<swid:regid>regid.1986-12.com.adobe</swid:regid>
</swid:software_creator>
<swid:software_licensor id="e8_3_5">
<swid:name>Adobe Systems Incorporated</swid:name>
<swid:regid>regid.1986-12.com.adobe</swid:regid>
</swid:software_licensor>
<swid:software_id id="e8_3_6">
<swid:unique_id>Photoshop-CS3-Win-GM-en_US</swid:unique_id>
<swid:tag_creator_regid>regid.1986-12.com.adobe</swid:tag_creator_regid>
</swid:software_id>
<swid:tag_creator id="e8_3_7">
<swid:name>Adobe Systems Incorporated</swid:name>
<swid:regid>regid.1986-12.com.adobe</swid:regid>
</swid:tag_creator>
<!-- Optional elements -->
<swid:elements_owner>
<swid:owner_regid>regid.1986-12.com.adobe</swid:owner_regid>
<swid:element_id>e8_3_1</swid:element_id>
<swid:element_id>e8_3_2</swid:element_id>
<swid:element_id>e8_3_3</swid:element_id>
<swid:element_id>e8_3_4</swid:element_id>
<swid:element_id>e8_3_5</swid:element_id>
<swid:element_id>e8_3_6</swid:element_id>
<swid:element_id>e8_3_7</swid:element_id>
<swid:element_id>e8_4_8</swid:element_id>
<swid:element_id>e8_4_20</swid:element_id>
<swid:element_id>e8_4_24</swid:element_id>
</swid:elements_owner>
<swid:license_linkage id="e8_4_9">
<swid:activation_status id="e8_4_9sub1">Licensed</swid:activation_status>
```

<swid:channel_type id="e8_4_9sub2">Retail</swid:channel_type>
<swid:customer_type id="e8_4_9sub3">Retail</swid:customer_type>
</swid:license_linkage>
<swid:serial_number id="e8_4_20">97078754260090944559975б</swid:serial_
number>
<swid:supported_languages id="e8_4_24">
<swid:language id="e8_4_24sub1">en</swid:language>
</swid:supported_languages>
<!-- Extended elements -->
<swid:extended_information>
<exti:install_state xmlns:exti="http://www.adobe.com/AdobeExtendedInfo.xsd" xsi:
schemaLocation="http://www.adobe.com/AdobeExtendedInfo.xsd AdobeExtInfo.xsd
">Installed</exti:install_state>
</swid:extended_information>
</swid:software_identification_tag>

Н.2 Пример 2

Пример 2: Mozilla Firefox
<?xml version="1.0" encoding="UTF-8"?>
<swid:software_identification_tag
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:swid="http://standards.iso.org/iso/19770/-2/2008/schema.
xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://standards.iso.org/iso/19770/-2/2008/schema.xsd software_identification_tag.xsd ">
<!-- Mandatory elements -->
<swid:entitlement_required_indicator>false</swid:entitlement_required_indicator>
<swid:product_title>Firefox</swid:product_title>
<swid:product_version>
<swid:name>2.0.0.10</swid:name>
<swid:numeric>
<swid:major>2</swid:major>
<swid:minor>0</swid:minor>
<swid:build>0</swid:build>
<swid:review>10</swid:review>
</swid:numeric>
</swid:product_version>
<swid:software_creator>
<swid:name>Mozilla Corporation</swid:name>
<swid:regid>regid.1994-11.com.mozilla</swid:regid>
</swid:software_creator>
<swid:software_licensor>
<swid:name>Mozilla Corporation</swid:name>
<swid:regid>regid.1994-11.com.mozilla</swid:regid>
</swid:software_licensor>
<swid:software_id>
<swid:unique_id>firefox.2.0.0.10</swid:unique_id>
<swid:tag_creator_regid>regid.1994-11.com.mozilla</swid:tag_creator_regid>
</swid:software_id>
<swid:tag_creator>
<swid:name>Mozilla Corporation</swid:name>
<swid:regid>regid.1994-11.com.mozilla</swid:regid>
</swid:tag_creator>
<!-- Optional elements -->
<swid:abstract lang="en">Corporation's flagship browser.</swid:abstract>
<swid:data_source>Electronic distribution</swid:data_source>
<swid:license_linkage>
<swid:activation_status>Fully licensed</swid:activation_status>
<swid:channel_type>Internet</swid:channel_type>
<swid:customer_type>End-user</swid:customer_type>
</swid:license_linkage>
<swid:product_category>
<swid:UNSPSC_ver>10.0501</swid:UNSPSC_ver>

```
<swid:segment_title>Information Technology Broadcasting and Telecommunications</swid:segment_title>
<swid:family_title>Software</swid:family_title>
<swid:class_title>Network applications software</swid:class_title>
<swid:commodity_title>Internet browser software</swid:commodity_title>
<swid:code>43232705</swid:code>
</swid:product_category>
</swid:software_identification_tag>
```

Н.3 Пример 3

Пример 3: Подписанный тег идентификации программного обеспечения

Примечание – В следующем примере используется example.com, стандартный домен для примеров IANA – издателя примеров приложений. В этом примере используется цифровая подпись, применяемая к элементам создатель программного обеспечения («software_creator»), лицензиар программного обеспечения («software_licensor»), создатель тега («tag_creator») и владелец элементов («elements_owner») тега идентификации программного обеспечения. После того как на элементах проставлена подписи, внесение изменений в эти элементы неизбежно повлечет за собой недействительнолсть цифровой подписи; тем не менее, другие элементы, содержащиеся в файле тега идентификации программного обеспечения, можно менять или дополнять без воздействия на действительность цифровой подписи.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<swid:software_identification_tag xmlns:swid="http://standards.iso.org/iso/19770/-2/2008/schema.xsd"
xmlns:ds="http://gost.e-ecolog.ru" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="gost.e-ecolog.ru software_identification_tag.xsd ">
<!-- Mandatory elements -->
<swid:entitlement_required_indicator xml:id="e8_3_1">true</swid:entitlement_required_indicator>
<swid:product_title xml:id="e8_3_2">Example Product</swid:product_title>
<swid:product_version xml:id="e8_3_3">
<swid:name xml:id="e8_3_3sub1">Ver 3.1</swid:name>
<swid:numeric xml:id="e8_3_3sub2">
<swid:major xml:id="e8_3_3sub2sub1">3</swid:major>
<swid:minor xml:id="e8_3_3sub2sub2">1</swid:minor>
<swid:build xml:id="e8_3_3sub2sub3">4</swid:build>
<swid:review xml:id="e8_3_3sub2sub4">1593</swid:review>
</swid:numeric>
</swid:product_version>
<swid:software_creator xml:id="e8_3_4">
<swid:name>Example Corp</swid:name>
<swid:regid>regid.1995-09.com.example</swid:regid>
</swid:software_creator>
<swid:software_licensor xml:id="e8_3_5">
<swid:name>Example Corp</swid:name>
<swid:regid>regid.1995-09.com.example</swid:regid>
</swid:software_licensor>
<swid:software_id xml:id="e8_3_6">
<swid:unique_id>Example Product-3.1.4.1593-en</swid:unique_id>
<swid:tag_creator_regid>regid.1995-09.com.example</swid:tag_creator_regid>
</swid:software_id>
<swid:tag_creator xml:id="e8_3_7">
<swid:name>Example Corp</swid:name>
<swid:regid>regid.1995-09.com.example</swid:regid>
</swid:tag_creator>
<!-- Optional elements -->
<swid:elements_owner xml:id="e8_4_6">
<swid:owner_regid>regid.1995-09.com.example</swid:owner_regid>
<swid:element_id>e8_3_1</swid:element_id>
<swid:element_id>e8_3_2</swid:element_id>
<swid:element_id>e8_3_3</swid:element_id>
<swid:element_id>e8_3_4</swid:element_id>
<swid:element_id>e8_3_5</swid:element_id>
<swid:element_id>e8_3_6</swid:element_id>
<swid:element_id>e8_3_7</swid:element_id>
```

```
<swid:element_id>e8_4_9</swid:element_id>
<swid:element_id>e8_4_23</swid:element_id>
</swid:elements_owner>
<swid:license_linkage xml:id="e8_4_9">
<swid:activation_status xml:id="e8_4_9sub1">Licensed</swid:activation_status>
<swid:channel_type xml:id="e8_4_9sub2">Volume</swid:channel_type>
<swid:customer_type xml:id="e8_4_9sub3">Educational</swid:customer_type>
</swid:license_linkage>
<swid:supported_languages xml:id="e8_4_23">
<swid:language xml:id="e8_4_23sub1">en</swid:language>
</swid:supported_languages>
<Signature xmlns="gost.e-ecolog.ru >
<SignedInfo><CanonicalizationMethod Algorithm="gost.e-ecolog.ru
gost.e-ecolog.ru ıts"/><SignatureMethod Algorithm="http://gost.e-ecolog.ru
gost.e-ecolog.ru ="#e8_3_4"><Transforms><Transform Algorithm="gost.e-ecolog.ru
xmldsig#enveloped- signature"/></Transforms><DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><
DigestValue>DLHgCey2snS6pLB5iwsfbPzy+Zw=</DigestValue>
</Reference><Reference URI="#e8_3_5"><Transforms><Transform Algorithm="http://gost.e-ecolog.ru
xmldsig#envelopedsignature"/></Transforms><DigestMethod Algorithm="gost.e-ecolog.ru
igestValue>ZVewVYYwUUgowacvSEtynFBQrac=</DigestValu e></Reference><Reference URI="#e8_3_6"><Transform
s><Transform
Algorithm="gost.e-ecolog.ru
Transforms><DigestMethod
Algorithm="gost.e-ecolog.ru >
aBr4ZdvIJ/Fxmml2xVd7oBmvAc8=</DigestValue><
/Reference><Reference URI="#e8_3_7"><Transforms><Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
jcya1UScQoyYlmZqIzrFI3y2tr0=</DigestValue></R eference><Reference URI="#e8_4_6"><Transforms><Transform
Algorithm="gost.e-ecolog.ru
xmldsig#enveloped-signature"/></Transforms><DigestMethod Algorithm=
"gost.e-ecolog.ru e></
Reference></SignedInfo>
<SignatureValue>f6dn2LyQIZwDLp123M9HQ/DYIVAy2S07zeZ23TOccpF2s+
KS2JQX6Q==</SignatureValue>
<KeyInfo>
<KeyValue>
<DSAKeyValue>
<P>/KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu0lmbzRMqzVDZkVG9xD7nN1ku
Fw==</P>
<Q>li7dzDacuo67Jg7mtqEm2TRuOMU=</Q>
<G>Z4Rxsnqc9E7pGknFFH2xqaryRPBaQ01khpMdLRQnG541Awtx/XPaF5Bpsy4pNWMOHCBiNU0NogpsQW5Q
vnlMpA==</G>
<Y>rJM4r2V+szYEH3v2Z9iwSztPdzbz4J870vz9yM7aqkYYXnQfS/vyIE24A/
DZOeNKUV16UTw5RQZOHpS94vCvfQ==</Y>
</DSAKeyValue>
</KeyValue>
</KeyInfo>
</Signature>
</swid:software_identification_tag>
```

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным стандартам Российской Федерации и действующим в этом качестве межгосударственным стандартам

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального, межгосударственного стандарта
ИСО 9001:2008	IDT	ГОСТ ISO 9001–2011 «Системы менеджмента качества. Требования»
ИСО/МЭК 33003	–	*
ИСО/МЭК 19770-1:2012	IDT	ГОСТ Р ИСО/МЭК 19770-1–2014 «Информационные технологии. Менеджмент программных активов. Часть 1. Процессы и оценка соответствия по уровням»
ИСО/МЭК 19770-5	–	*

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

Примечание – В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов: - IDT – идентичные стандарты.

Библиография

[1] ИСО/МЭК 19770-1:2006 Информационные технологии. Управление программными активами. Часть 1. Процессы (ISO/IEC 19770-1:2006, Information technology – Software asset management – Part 1: Processes)

[2] ИСО/МЭК 20000-1:2005 Информационные технологии. Управление услугами. Часть 1. Спецификации (ISO/IEC 20000-1:2005, Information technology – Service management – Part 1: Specification)

[3] ИСО/МЭК 20000-2:2005 Информационные технологии. Управление услугами. Часть 2. Нормы и правила (ISO/IEC 20000-2:2005, Information technology – Service management – Part 2: Code of practice)

[4] ИСО/МЭК 25051:2006 Проектирование программного обеспечения. Требования и оценка качества программных продуктов (SQuaRE). Требования к качеству готового к использованию коммерческого программного продукта (COTS) и инструкции по тестированию (ISO/IEC 25051:2006, Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing)

[5] IETF RFC 1034 Доменные имена. Концепции и устройства (IETF RFC 1034, Domain names – Concepts and facilities)

[6] IETF RFC 1123 Требования к хостам сети Интернет. Применение и поддержка (IETF RFC 1123, Requirements for Internet Hosts – Application and Support)

[7] IETF RFC 4647 Сопоставление языковых тегов (IETF RFC 4647, Matching of Language Tags)

УДК 65.012:004.45:006.354 ОКС 35.080 П85 ОКСТУ 4090

Ключевые слова: информационные технологии, программные активы, процессы, оценка соответствия

Редактор *Е.А. Севко*
Технический редактор *А.Б. Заварзина*
Корректор *В.Г. Смолин*
Компьютерная верстка *Д.Е. Першин*

Сдано в набор 24.09.2015. Подписано в печать 25.10.2015. Формат 60x841/8 Гарнитура Ариал.
Усл. печ. л. 11,16. Уч.-изд. л. 9,93. Тираж 36 экз. Зак. 3356.

Набрано в ООО «Академиздат».
www.academizdat.com gost.e-ecolog.ru

Издано и отпечатано во
ФГУП «СТАНДАРТИНФОРМ». 123995 Москва, Гранатный пер., 4.
www.gostinfo.ru gost.e-ecolog.ru

gost.e-ecolog.ru