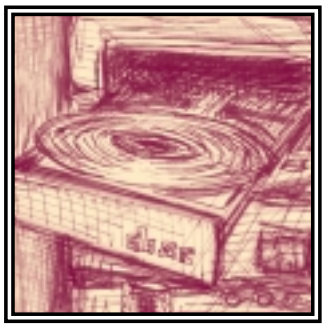

OLE for Process Control – свобода выбора

Дмитрий Теркель

В статье рассматривается OLE for Process Control (OPC) — основной стандарт взаимодействия между программными компонентами современных систем сбора данных и управления (SCADA). Обсуждаются основные концепции стандарта, а также вопросы производительности и разработки OPC-серверов.

Что такое OPC?

OPC (OLE for Process Control) – это стандарт взаимодействия между программными компонентами системы сбора данных и управления (SCADA), основанный на объектной модели COM/DCOM фирмы Microsoft. Через интерфейсы OPC одни приложения могут читать или записывать данные в другие приложения, обмениваться событиями, оповещать друг друга о нештатных ситуациях (тревогах), осуществлять доступ к данным, зарегистрированным в архивах (так называемые «исторические» данные). Эти приложения могут располагаться как на одном компьютере, так и быть распределенными по сети, при этом независимо от фирмы-поставщика стандарт OLE for Process Control, признанный и поддерживаемый всеми ведущими фирмами-производителями SCADA-систем и оборудования, обеспечит их совместное функционирование. Особый класс OPC-приложений представляют собой OPC-серверы конкретных аппаратных устройств – они поставляются многими производителями аппаратуры (а также независимыми производителями, но в этом случае они, как правило, не бесплатные). OPC-сервер создает своего рода абстракцию аппаратуры, позволяя любому OPC-клиенту записывать и считывать данные с устройства. Устройство, для которого есть OPC-сервер, может использоваться вместе с любой современной SCADA-системой.

Теперь у пользователя – системного интегратора или разработчика АСУ ТП — появилась возможнось выбирать оптимальные для своей системы компоненты, а не ориентироваться, как раньше, целиком на «монолитные» решения, предлагаемые тем или иным поставщиком. OPC – это свобода выбора.

Что не может OPC

Конечно, не бывает средств, решающих сразу все проблемы. OPC может использоваться только там, где установлен Microsoft DCOM, а это на сегодня Windows NT, Windows 95/98 и теоретически некоторые системы семейства Unix. В Windows CE DCOM (от Microsoft) в настоящее время отсутствует, однако попытки перенести OPC на эту платформу предпринимаются (предлагается некий паллиатив DCOM). «Глубже» Windows CE – во встроенные контроллеры и PLC – OPC вряд ли опустится. В целом OPC – это интерфейс для системы верхнего уровня. Ниже лежащие слои – PLC, УСО и т.д. – представлены для нее в виде OPC-серверов и в общем случае являются «черными ящиками».

Далее, OPC не обеспечивает работы в жестком реальном времени (пока, во всяком случае), поскольку в DCOM отсутствуют понятия качества обслуживания, крайних сроков и т.п. В то же время контроль за «устареванием» данных имеется: каждое передаваемое значение (тег) сопровождается меткой времени происхождения (timestamp). Несмотря на то, что требования жесткого реального времени, строго говоря, не выполняются, реальная частота передачи данных порядка 50 миллисекунд достигается без каких-либо специальных мер.

Не следует думать, что любое устройство можно просто так «через OPC» подключить к любой SCADA-системе, – для этого надо иметь OPC-сервер. Сервер можно либо получить вместе с устройством, либо купить, либо написать самостоятельно. Последнее не так уж сложно — и это, как правило, является наилучшим рецептом при модернизации уже сложившихся систем управления, где используется нестандартная аппаратура. Так можно получить вполне «современное» лицо системы (используя, например, SCADA-систему Genesis32 фирмы Iconics), сохранив при этом проверенный аппаратный и даже программный задел (написанные ранее алгоритмы управления можно просто встроить в разрабатываемый OPC-сервер).

Как это работает

Реализация OPC основана на объектной модели COM/DCOM фирмы Microsoft. COM — это Component Object Model – модель многокомпонентных

объектов, позволяющая приложению манипулировать удаленными программными объектами, точнее, вызывать те или иные функции (методы) этих объектов так, как будто объекты находятся «рядом». Объект может находиться и в самом деле рядом (в адресном пространстве приложения) – тогда это просто COM.

Если же объект находится в другой программе на том же компьютере или на другом узле сети, то это DCOM – Distributed (распределенная) COM. В распределенном случае (DCOM) вызов любой функции объекта перехватывается специальным агентом-посредником, так называемой proxy/stub DLL, которая выполняет роль представителя объекта у обратившегося к нему клиента.

Proxy/stub DLL упаковывает параметры функции (marshaling - транспортировка) и передает вызов операционной системе, которая (возможно, по сети) доставляет вызов по назначению, то есть заставляет реальный объект выполнить заданную функцию. Результат затем возвращается (примерно по той же цепочке) приложению-клиенту. Удобство использования DCOM состоит в том, что приложение-клиент совершенно не обязано знать, где реально находится объект – о степени удаленности объекта оно может судить только по увеличению расхода времени на вызов функции.

OPC-взаимодействие основано на клиент-серверной схеме. OPC-клиент (например, SCADA), вызывая определенные функции объекта OPC-сервера, подписывается на получение определенных данных с определенной частотой. В свою очередь, OPC-сервер, опросив физическое устройство, вызывает известные функции клиента, уведомляя его о получении данных и вручая сами данные. Таким образом, при OPC-взаимодействии используются как прямые COM-вызовы (от клиента к серверу), так и обратные (callback, от сервера к клиенту). Это надо учитывать при настройках безопасности DCOM в Windows NT: если клиент «видит» данные, но не получает их, значит, скорее всего, система безопасности NT блокировала обратные вызовы.

Стандарт OPC, в отличие, например, от устаревшего DDE (Dynamic Data Exchange), хотя и основан на универсальном фундаменте – COM/DCOM, разрабатывался специально для использования в промышленной автоматизации, поэтому он имеет вполне содержательную концептуальную сторону, то есть, на самом деле, свою проблемно-ориентированную модель взаимодействия, которая и реализована через совокупность COM-интерфейсов. Эта концептуальная сторона в известной степени независима и представляет самый большой интерес, особенно для пользователя-непрограммиста, для которого тонкости реализации COM-интерфейсов не столь важны. В принципе, основные идеи OPC могли бы быть реализованы и с помощью других объектных технологий (получилось бы что-нибудь вроде CORBA for Process Control, например), однако распространенность Windows-платформ предопределила выбор в пользу стандартов Microsoft.

Концепция стандарта OPC

Стандарт состоит из трех основных спецификаций:

1) доступ к данным реального времени (Data Access);
2) обработка тревог и событий (Alarms & Events);
3) доступ к историческим данным (Historical Data Access).

OPC-серверов, соответственно, тоже может быть три вида, хотя не возбраняется совмещать все эти функции в одном. OPC-серверы физических устройств обычно являются только серверами данных (Data Access Servers). Серверы тревог и исторические чаще всего «паразитируют» на серверах данных. Сервер тревог формирует определенные логические переменные, называемые состояниями (conditions), имея в качестве исходной информации некую переменную (тег), полученную от сервера данных. Состояния изменяют свое значение, если переменная, например, вышла за допустимые границы. Об изменении состояния сервер тревог оповещает клиентов, посылая им событие (тревогу), а клиент возвращает серверу подтверждение, что он тревогу воспринял. Впрочем, могут существовать состояния, не связанные с каким-либо параметром и управляемые сервером тревог по собственному усмотрению (например, если сервер тревог напрямую взаимодействует с аппаратурой, он может устанавливать или снимать состояние неисправности). Серверы исторических данных получают от серверов данных параметры в реальном времени и архивируют их, а затем предоставляют эти данные другим приложениям (например, для построения графиков трендов).

Центральное место среди спецификаций OPC занимает доступ к данным реального времени (DataAccess). Это самая старая и отработанная спецификация, в настоящее время действует ее вторая версия.

Базовым понятием этой спецификации является элемент данных (Item). Каждый элемент данных (то есть фактически – параметр технологического процесса) имеет значение, время последнего обновления (timestamp) и признак качества, определяющий степень достоверности значения. Значение может быть практически любого скалярного типа: булево, целое, с плавающей точкой и т.п. – или строкой (на самом деле это так называемый OLE VARIANT). Время представляется с 100-наносекундной точностью (на самом деле это FILETIME Win32 API). Качество — это код, содержащий в себе грубую оценку достоверности параметра — UNCERTAIN, GOOD и BAD (не определено, хорошее и плохое), а на случай плохой оценки — еще и расшифровку, например, QUAL_SENSOR_FAILURE – неисправность датчика.

Следующим вверх по иерархии является понятие группы элементов (OPC Group). Группа создается OPC-сервером по требованию клиента, который затем может добавить в группу элементы (Items). Для группы клиентом задается частота обновления данных, и все данные в группе сервер старается обновлять и передавать клиенту с заданной частотой. Отдельно стоящих вне группы элементов быть не может. Клиент может создать для себя на сервере несколько групп, различающихся требуемой частотой обновления. Группа (кроме так называемых публичных групп) всегда создается для каждого клиента своя, даже если состав элементов и частоты обновления совпадают. Отсоединение клиента приводит к уничтожению созданных для него групп.

Элементы в группе – это своего рода клиентские ссылки на некие реальные переменные (теги), находящиеся на сервере или в физическом устройстве. Понятие тега спецификацией OPC не определяется, но подразумевается неявно. Элементы в группу клиент добавляет по имени, и эти имена, разумеется, на самом деле являются именами соответствующих тегов. Клиент может либо знать нужные имена заранее (если он такой умный), либо запросить список имен тегов у сервера.

Для запроса имен тегов служит интерфейс IOPCBrowseServerAddressSpace, с помощью которого сервер описывает клиенту свое «пространство имен», организованное в общем случае иерархически. Пример полного имени тега: `Устройство_1. Модуль_2. Аналоговый Вход_3` (в качестве разделителя используется

точка). При добавлении элемента в группу клиент всегда указывает это полное имя. Заметим, что группы, создаваемые клиентом на сервере, не обязаны совпадать (и, как правило, не совпадают) с подразделами пространства имен сервера, элементы в группу добавляются вразнобой. Единственное, что их объединяет, – это общая частота обновления и синхронность отправки клиенту.

Наконец, на верхней ступеньке иерархии понятий находится сам OPC-сервер. Из всех перечисленных (OPC-группа, OPC-элемент) он единственный является COM-объектом, все остальные объекты доступны через его интерфейсы, которые он предоставляет клиенту.

Схема взаимодействия OPC-клиентов с OPC-сервером представлена на рисунке.

Обмен данными между клиентом и сервером может осуществляться в двух режимах — синхронном и асинхронном. При асинхронном варианте обмена сервер сам оповещает клиента об изменившихся значениях данных, на которые подписался клиент (по возможности с частотой, заданной клиентом при создании группы). При синхронном клиент осуществляет инициативное чтение или запись данных. Запись может быть только синхронной.

Производительность

Затраты ресурсов компьютера на поддержку OPC-взаимодействия и предельно достижимая интенсивность обмена между клиентом и сервером зависят от ряда факторов, но прежде всего – от того, находится ли сервер непосредственно в адресном пространстве клиента (так называемый внутризадачный — InProcess Server) или он является самостоятельным приложением. В последнем случае важно, расположен сервер на том же компьютере, что и клиент, или на другой станции локальной сети. Третий по важности фактор – это возможность группировки данных для отправки клиентам. Так, передача раз в 1 секунду 100 тегов займет значительно меньше ресурсов, чем одного тега через каждые 10 миллисекунд.

Ориентировочные экспериментальные данные о пропускной способности интерфейсов OPC опубликованы на Web-узле OPC Foundation — www.opcfoundation.org (там же можно получить более подробное описание условий проведения эксперимента).

Предельная пропускная способность внутризадачного сервера может достигать (здесь и далее – на компьютере с процессором Pentium-233) 1 млн. элементов OPC в секунду, что значительно превышает все мыслимые реальные потребности. Однако не все OPC-серверы имеют внутризадачный (InProcess) вариант – для этого они должны оформляться как динамические библиотеки (DLL), а не как самостоятельные программы.

В случае локального сервера (отдельная программа, но на том же компьютере) пропускная способность колеблется примерно от 3 до 60 тысяч элементов в секунду. Наихудший вариант соответствует передаче единственного элемента с максимальной частотой, наилучший – передаче одновременно 100 или более элементов с соответственно меньшей частотой.

Наконец, для удаленного сервера (в сети Ethernet 10Base–T) пропускная способность оказалась заключена в пределах 330 – 7000 элементов в секунду (наилучший и наихудший варианты соответствуют тем же самым крайним случаям).

Не следует придавать приведенным цифрам абсолютное значение – они могут варьироваться в зависимости от системной конфигурации, а также особенностей реализации сервера и клиента. Так, по данным Rockwell Automation, OPC-сервер (отдельное приложение, не внутризадачный) может обеспечить обмен с сетевыми клиентами до 200000 элементов в секунду. С другой стороны, столь оптимистические оценки могут быть резко снижены за счет ресурсоемкости обмена с реальной аппаратурой – коммуникационными портами, интерфейсными модулями и т.п., что, конечно, не учитывалось в тестовых экспериментах, поскольку не имеет прямого отношения к OPC. С точки зрения программной реализации, важно, насколько сервер использует многопоточность (multithreading), в частности, для обслуживания различных групп, а также производится ли кэширование значений тегов с целью минимизации количества запросов к аппаратуре.

Схема взаимодействия OPC-клиентов с OPC-сервером

Процесс стандартизации

Стандарт OPC разрабатывает независимая организация – OPC Foundation, насчитывающая более 170 членов, среди которых Siemens, Fisher-Rosemount, Honeywell, Rockwell, Iconics и др., то есть практически все известные (и не очень известные) компании-производители SCADA-систем и оборудования для систем промышленной автоматизации. Техническая деятельность OPC Foundation осуществляется в рабочих группах по направлениям. Среди этих направлений:

- доступ к данным реального времени OPC (Data Access Working Group);
- обработка тревог и событий (OPC Alarms and Events Working Group);
- защита данных (OPC Security Working Group);
- подтверждение соответствия стандартам OPC (OPC Compliance Working Group);
- доступ к историческим (архивным) данным (OPC Historical Working Group);
- Windows CE (OPC Windows CE Working Group).

К настоящему моменту выпущена вторая версия спецификации (версия 2.0), определяющая интерфейс доступа к данным реального времени (октябрь 1998 г.). До этого действовала спецификация 1a, которая продолжает поддерживаться OPC-клиентами (то есть серверы, разработанные в соответствии со

старой спецификацией, можно продолжать использовать без каких-либо доработок). Версия 2.0 отличается несколько измененным интерфейсом, используемым при соединении с клиентом (IConnectionPoint), улучшенной поддержкой национальных языков (теперь клиент может запросить сервер, какие языки он поддерживает, и выбрать среди них язык общения) и некоторыми другими модификациями. Однако концептуальная сторона, описанная ранее, осталась прежней.

Относительно недавно появилась первая спецификация, касающаяся обработки тревог и событий (Alarms & Events), она также доступна с Web-узла OPC Foundation. Спецификация доступа к историческим данным на момент написания статьи все еще не была окончательно утверждена, но, видимо, должна появиться в ближайшее время.

Как разработать свой OPC-сервер

Утвержденные спецификации OPC, а также proxy/stub DLL, соответствующая интерфейсам OPC, свободно доступны с Web-узла OPC Foundation. Строго говоря, этого достаточно для разработки своего OPC-сервера. Однако разработка сервера «с нуля» — довольно сложный процесс, а аккуратная реализация интерфейсов OPC в многопоточной (multithread) среде изобилует различными «подводными камнями». Это представляет известную опасность, так как OPC-сервер обязан быть достаточно надежной программой.

Проще всего разработать сервер, используя специально созданные для этого средства. Так, фирма Iconics, известная своей SCADA-системой Genesis, предлагает OPC ToolWorX, который оформлен в виде дополнительного мастера (Wizard), встраиваемого в среду разработки Visual C++. Мастер генерирует некий «образцовый» проект, в котором требуется только модификация фрагментов кода, связанных со спецификой обслуживаемого устройства или протокола. Поддержка OPC-взаимодействия обеспечивается специальными классами объектов, не требующими каких-либо исправлений, поэтому программист может сосредоточиться на функциональности своего устройства, не заботясь о реализации собственно OPC-интерфейсов. Все компоненты OPC ToolWorx поставляются в исходном коде. Для квалифицированного программиста срок разработки несложного OPC-сервера с помощью Iconics OPC ToolWorX – 3-4 недели.

В случае, если к серверу не предъявляются особо жесткие требования к синхронности обновления данных и не требуется динамическая реструктуризация пространства имен во время его работы, можно применить самое простое и недорогое решение – универсальный OPC-сервер фирмы Fastwel. Он создан на базе Iconics ToolWorX и предусматривает подключение динамической библиотеки (DLL), написанной пользователем, в которой сосредоточен весь код, специфичный для обслуживаемого устройства. Интерфейс этой DLL с сервером очень прост, и разработка ее для простых устройств (или когда уже есть соответствующий программный задел в виде ранее написанных драйверов и т.п.) занимает у квалифицированного программиста всего 1-3 дня. Вместе с универсальным OPC-сервером (в виде исполняемого модуля) поставляется исходный текст «образцовой» DLL, который можно использовать как пример реализации.

Достаточно широко известен также пакет для разработки OPC-приложений фирмы Intellution. Для разработчиков АСУ ТП, использующих SCADA-систему FiX Dynamics, это, безусловно, наилучший выбор. Однако для написания «просто OPC-сервера» пакет от Intellution чрезмерно перегружен «фирменными» чертами: по существу, разрабатывается не сервер как таковой, а FiX-драйвер, открытый остальному миру через OPC-интерфейсы. Впрочем, выбор инструментальных средств разработки — во многом вопрос вкуса и привычки.

SCADA-системы и OPC

Как уже говорилось, практически все производители SCADA-систем поддерживают OLE for Process Control, среди них – FixDynamics (Intellution), FactorySuite 2000 (Wonderware), Genesis32 (Iconics), FactoryLink(US Data), Lookout (National Instruments) и т.д. Однако для большинства из них OPC — только один из поддерживаемых интерфейсов взаимодействия, их внутренняя идеология построения напрямую не связана с этим стандартом и часто унаследована от предыдущих 16-разрядных версий. По-видимому, пока только Genesis32 изначально спроектирован на основе OPC (OPC to the Core). Все компоненты Genesis32 взаимодействуют между собой через OPC, являясь, в зависимости от ситуации, либо серверами, либо клиентами, либо и теми и другими одновременно. Это придало системе во многом единый стиль, стройность архитектуры и предопределило ее компонентный характер: унификация интерфейсов взаимодействия дала возможность легко выбирать совокупность компонентов, действительно необходимых пользователю (и не переплачивать за ненужные). Конечно, у других SCADA-систем есть, безусловно, свои сильные стороны, однако Genesis32 в наибольшей степени дает своим пользователям почувствовать свободу выбора, которую обеспечивает OLE for Process Control. ●