Executive Summary

Data in today's enterprise lives in silos. The historian speaks a different language than the ERP system which speaks a different language than the maintenance system and so on. If an effective operations intelligence solution requires pieces of information from each of these systems, how do we do it? Historically the best option was to copy all of the data from the silos to yet another repository. This required all parties to agree on a single universal integration schema, which is a costly proposition.

The Intuition™ semantic model provides a "virtual repository" where data from the isolated silos is stitched together on demand. Simply put, the data stays where it is – in its unique format – while Intuition performs the acrobatics required to federate data from the source systems. This fresh approach supports the proposed service and component structure of the Microsoft Upstream Reference Architecture (MURA), an IT architecture that serves as a common, reliable environment for the implementation and integration of the many technologies that make up the digital oil field. Ultimately, this architecture will help to dramatically improve efficiency and cost-effectiveness for upstream oil and gas analysis, operations, and business.

Intuition is a family of software and technologies for the process industries. The Intuition family provides site and enterprise access to production and operation information, in context, and enables fast and effective advanced visualization and integration of applications for managing, monitoring and optimizing operations. The Intuition semantic model is a cornerstone of the Intuition family; it provides users with seamless access to a broad range of data sources, without requiring duplication of data or change to source systems.

In all, Intuition promotes a lower total cost of ownership throughout the solution lifecycle by better supporting rapid deployment and a lower operating expenditure within the federated data environments typical of the Operational, Production and Asset Management functions.

Jay Funnell, Principal Architect, Honeywell
in collaboration with:
David Dickson, Principal Industry Consultant, Honeywell
Joel Chacon, Solution Consultant, Honeywell
Why use a Semantic Model?

A Canadian, a Venezuelan, and an Australian get into a car…

No, this isn’t the opening line to a rude joke but a true story. It happened to me while en route to a client site where we were to host a demonstration of Intuition. There was the usual ebb and flow of traffic as our driver adeptly navigated the busy streets of Brisbane when suddenly a peculiar car drove past.

The Canadian said, “Look it’s a Ranchero”. The Australian said, “That’s a Ute”. And not to be left out, the Venezuelan chimed in, “No, no, El Camino!”

We grew up in different surroundings yet had a shared understanding of the vehicle that looked part car, part truck. We shared our individual experiences with this vehicle with three different names and did so without much formality. There was a kind of effortless exchange of knowledge that was no more than what our primal ancestors would have used when comparing notes about the smelly sabre tooth tiger that was hiding behind a rock.

This is the nature of semantic modeling. Humans intuitively describe, categorize, compare, and contrast things that we encounter with our senses. When we talk to others a kind of handshake takes place between minds that allows us to exchange ideas while simultaneously increasing our own understanding of the world. It seems counter-intuitive to first identify all possible aspects of a thing before speaking of it, yet this is exactly what traditional modeling tools that subscribe to the Closed World Assumption ask of us.

The Traditional Approach

To illustrate the point, let’s describe the half car, half truck using a traditional relational model. We’ll expand the conversation and use it as our guide. Next, we’ll contrast it to the process of creating a semantic model.

| Canadian: | Look it’s a Ranchero |
| Australian: | That’s a Ute |
| Venezuelan: | No, no, El Camino! |

| Canadian: | I remember seeing those back in 1979. They were part car and part truck. I saw a blue one that was made by Ford. That can’t be a Ute though, aren’t those farm vehicles? |
| Australian: | No, that’s a Sports Ute. They are popular with the younger guys. The classic Holden Ute is the one you’re thinking about. |
| Venezuelan: | The El Camino was very similar. It was a two-seater and some had V8 engines. |

The Relational Model

1. We start with an existing set of data that is unique to each person’s experiences, much like the data silos in an enterprise. Imagine that each table is stored in a separate repository to mimic an enterprise environment.

| CREATE TABLE Ranchero (Year int, Color varchar(255), Manufacturer varchar(255)) |
| CREATE TABLE Ute (Year int, Manufacturer varchar(255) IsSportyUte bool) |
| CREATE TABLE El-Camino (Year int, Seats int, Engine varchar(255)) |
2. Next, let’s say that we need a report that contains a summary of all known half car, half truck entities in the enterprise. Our data is spread across the El Camino, Ranchero, and Ute repositories yet they describe the same thing – a “half car, half truck.” In an Upstream oil and gas environment we might be dealing with ten to fifteen repositories or sources of master data – perhaps a well data historian, well maintenance system, geological survey database, production database, environmental & compliance system, ERP system, etc. How do we do it? There are several options.

   a. **Keep the data isolated**
      - As the path of least resistance, this is the current reality for most enterprises. It essentially ignores integration altogether.
      - Each repository has a separate user interface and data is not shared. There is no single place where someone can “at a glance” obtain a summary of key performance indicators (KPIs) that span data sources. This is a significant barrier to becoming an agile enterprise.

   b. **Leave it up to the reports**
      - Each report queries each data source separately then merges the data before presenting it.

      The key problem with this approach is that:
      - each report must be intimately aware of the data structures in each repository
      - each report must also know how to harmonize the data between silos
      - each new report duplicates the effort already expended by the previous report builders

      Clearly this is not an efficient solution.

   c. **Create a data warehouse, introduce a point-to-point integration project, and synchronize the data**
      - With the relational method, we need to unify these concepts into a common table so that we can create queries, reports, etc.

      ```sql
      CREATE TABLE HalfCarHalfTruck
      (      
        CarName varchar(255),
        Year int,
        Seats int,
        Engine varchar(255),
        Manufacturer varchar(255),
        IsSportyUte bool,
        Color varchar(255)
      )
      ```

      The key challenges of this method include:
      - How recent is the data? For historians and real-time data sources, the synchronization process must run constantly to ensure that the warehouse is up to date. This is not an effective use of corporate bandwidth and places a huge load on the operational data sources. Therefore in practice, the information in data warehouses is often out of date.
      - What if a new column is added to the El-Camino, Ranchero, or Ute tables to describe a different aspect? The HalfCarHalfTruck table won’t know about it so we must go back to the data warehouse and add a new column then tell it where to get the data. This is costly from a maintenance perspective.
      - Data is duplicated around the enterprise to the warehouse; again introducing master data governance overheads and headaches.

      None of these solutions are ideal.
The Semantic Approach
When we saw the half car, half truck that day, we didn’t have to pull over and draw up a schema in a Master Data Management system to talk about it. Instead we simply talked about it based on our diverse yet imperfect knowledge, similar to the Semantic approach used in Intuition.

Evolution of Semantic Modeling
Each system contains some model information: models of laboratory data, maintenance data, inventory, and so on. These models will nearly always be different—they address different problems, have different users, may have been designed at different times, or may have come from different companies. Integrating these systems requires you to integrate the models. Relational and object integration approaches often replicate the data. Even if they can avoid replicating data, they still must replicate ‘model’ information. This immediately becomes a maintenance problem. The semantic database, on the other hand, is built on the same principle that treats the entire Internet as one large, federated database. This allows queries that span many external data sources without the need to replicate the information. For example, Tank101’s disposition (the material in the tank) is in the scheduling database, the specification of the material is in the laboratory database, and the outstanding maintenance work orders are in the maintenance database. All of these can be federated into a single report about that tank without the need to replicate data. Better still, the semantic modeling approach does not require replicating the models for this federation to work. This significantly lowers the Total Cost of Ownership by offering a more rapidly deployable solution and reducing ongoing operating expenditures.

Intuition stores the master plant model in a semantic database structure. It combines a simple structure with the ability to express complex relationships. Whereas a relational database relies on referential integrity to control the contents of the database, a semantic database uses ‘rules’ that are verified by the built-in inference engine. Relational databases, object databases and real-time historians are great at storing information, but they have limited or no intelligence. The semantic database is built on artificial intelligence principles, thus allowing intelligence to be gathered from the information.

For example, if FI101 is the flow of Pump101, and Pump101 is upstream of Tank10, then the flow into Tank10 can be inferred as FI101 directly from the semantic database: no programs, no duplicate data, just intelligence.
Or: if it has four wheels like a Ute; is the size of a Ute; has a tray like a Ute; then it is a Ute.

Let’s walk through the same conversation and see how the semantic approach works.

<table>
<thead>
<tr>
<th>Canadian: Look it’s a Ranchero</th>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australian: That’s a Ute</td>
<td>thing1</td>
<td>rdf:type</td>
<td>Ranchero</td>
</tr>
<tr>
<td>Venezuelan: No, no, El Camino!</td>
<td>thing1</td>
<td>rdf:type</td>
<td>Ute</td>
</tr>
<tr>
<td></td>
<td>thing1</td>
<td>rdf:type</td>
<td>El Camino</td>
</tr>
</tbody>
</table>

Conceptually, a semantic database is one table. Sounds too simple, right? The key is to reduce the metadata to simple statements that contain a subject, predicate, and object. This has its roots in the W3C RDF standard. For this example we’ll use the resource called “thing1” to identify the half car, half truck that was spotted.

We did not create a new table. In fact, we are just populating a virtual table of statements. We don’t even need to fully define the attributes and relationships at this time. At this point we know that there is a thing called “thing1” and that it has three names.

<table>
<thead>
<tr>
<th>Canadian: I remember seeing those back in 1979. They were part car and part truck. I saw a blue one that was made by Ford. That can’t be a Ute though; aren’t those farm vehicles?</th>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ranchero</td>
<td></td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ute</td>
<td></td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>El Camino</td>
<td></td>
</tr>
<tr>
<td>Ranchero</td>
<td>wasMadeIn</td>
<td>1979</td>
<td></td>
</tr>
<tr>
<td>Ranchero</td>
<td>rdf:type</td>
<td>car</td>
<td></td>
</tr>
<tr>
<td>Ranchero</td>
<td>rdf:type</td>
<td>truck</td>
<td></td>
</tr>
<tr>
<td>blueThing</td>
<td>rdf:type</td>
<td>Ranchero</td>
<td></td>
</tr>
<tr>
<td>blueThing</td>
<td>hasManufacturer</td>
<td>ford</td>
<td></td>
</tr>
</tbody>
</table>

We’ve now added additional statements to the semantic database. This time we’re expanding on the thing called “Ranchero.”

<table>
<thead>
<tr>
<th>Australian: No, that one’s more of a sporty Ute. They’re popular with the younger guys. The classic Holden Ute is the one you’re thinking about.</th>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ranchero</td>
<td></td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ute</td>
<td></td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>El Camino</td>
<td></td>
</tr>
<tr>
<td>Ranchero</td>
<td>wasMadeIn</td>
<td>1979</td>
<td></td>
</tr>
<tr>
<td>Ranchero</td>
<td>rdf:type</td>
<td>car</td>
<td></td>
</tr>
<tr>
<td>Ranchero</td>
<td>rdf:type</td>
<td>truck</td>
<td></td>
</tr>
<tr>
<td>blueThing</td>
<td>rdf:type</td>
<td>Ranchero</td>
<td></td>
</tr>
<tr>
<td>blueThing</td>
<td>hasManufacturer</td>
<td>Ford</td>
<td></td>
</tr>
<tr>
<td>sportyUte</td>
<td>owl:subClassOf</td>
<td>Ute</td>
<td></td>
</tr>
<tr>
<td>classicUte</td>
<td>owl:subClassOf</td>
<td>Ute</td>
<td></td>
</tr>
<tr>
<td>sportyUte</td>
<td>avgDriverAge</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

Now we have the additional knowledge about Utes.
**Venezuelan**: The El Camino was very similar. It was a two-seater and some had V8 engines.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ranchero</td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ute</td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>El Camino</td>
</tr>
<tr>
<td>Ranchero</td>
<td>wasMadeIn</td>
<td>1979</td>
</tr>
<tr>
<td>Ranchero</td>
<td>rdf:type</td>
<td>car</td>
</tr>
<tr>
<td>Ranchero</td>
<td>rdf:type</td>
<td>truck</td>
</tr>
<tr>
<td>blueThing</td>
<td>rdf:type</td>
<td>Ranchero</td>
</tr>
<tr>
<td>blueThing</td>
<td>hasManufacturer</td>
<td>Ford</td>
</tr>
<tr>
<td>sportyUte</td>
<td>owl:subClassOf</td>
<td>Ute</td>
</tr>
<tr>
<td>classicUte</td>
<td>owl:subClassOf</td>
<td>Ute</td>
</tr>
<tr>
<td>sportyUte</td>
<td>avgDriverAge</td>
<td>25</td>
</tr>
<tr>
<td>El Camino</td>
<td>hasSeats</td>
<td>2</td>
</tr>
<tr>
<td>El Camino</td>
<td>hasEngine</td>
<td>V8</td>
</tr>
<tr>
<td>El Camino</td>
<td>owl:sameAs</td>
<td>sportyUte</td>
</tr>
<tr>
<td>El Camino</td>
<td>owl:sameAs</td>
<td>Ute</td>
</tr>
</tbody>
</table>

We now have a set of statements from which to share knowledge. From the sub-context of the conversation we’re going to assert that the Ranchero, Ute, and El Camino are all half-car, half-truck vehicles. This is accomplished using an OWL “sameAs” predicate.

The power of the semantic model is that it allows us to harmonize data from different repositories. An Oil & Gas “well” might be called an “asset” in an ERP system, a “producer” in a relational database, and a collection of “tags” in a historian. We can use the notion of sameAs to assert equality.

In case you are wondering: the “rdf” and “owl” prefixes are W3C standards. OWL is used to define ontology. To understand what ontology is we can refer to the following snippet from the OWL standard by the W3C:

> “OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology.”
Let’s take a look at some common queries. For example, let’s say a Canadian wants to know what things are like a (of type) “Ranchero” and what they are called; we can write a query like this.

```
SELECT ?thing
WHERE
{ ?thing rdf:type Ranchero }
```

The data is left in the original tables and federated at query time. This allows the report writer to use the vocabulary that is most comfortable.

Another common request is to find out everything that we know about a particular thing. Unlike a master data schema, this can be quite fluid in a semantic model. Progressing from the query above, now we want to know everything that there is to know about “thing1”.

```
SELECT ?predicate ?object
WHERE
{ thing1 ?predicate ?object }
```

This simply asks the system, “What do you know about thing1?” The response tells us that thing1 can be described as a Ranchero, Ute, or an El Camino. Next we can ask what is known about these types.

```
SELECT ?property ?value
WHERE
{ ?x ?property ?value .
  FILTER( ?x = El Camino || ?x = Ranchero || ?x = Ute)
}
```

We could also ask questions based on the relationships we created. “What vehicles are typically driven by someone in their twenties?”

```
SELECT ?vehicleType
WHERE
{ ?vehicleType avgDriverAge ?age .
  FILTER( ?age > 19 && ?age < 30) .
}
```

Semantic models are an excellent approach to data federation for a number of reasons:

- They embrace diversity. Traditional master data management techniques stifle diversity by forcing conformity with a master schema.
- They can apply equality to similar concepts from separate ontology, such as facts found about the same entity (“a pump” or “a well”) in different systems (historian, ERP, CMMS).
- They can be built incrementally as information becomes available. Quite a nice feature to have in the deployment of real world systems!
- They are ‘lightweight’ and ‘flexible’.
- The Intuition semantic model leaves data in its original repository and “federates” it on demand; a significant benefit that drastically reduces master data management headaches.
Conclusion

So what can you do with a semantic model that you can't do with other models?

You can make a mistake without losing your job.

Traditional relational meta-models have a great deal of flexibility...up until they are put into production. Once data lands in a table and reports are built, it becomes difficult to change the tables and entity relationships. Those early decisions about table structure, keys, and data types will influence every future decision for applications. Semantic models allow us to define structure whenever necessary. They also allow us to infer structure at runtime. Semantic models are flexible and adept at handling complexity and managing change. This is especially important for expanding production environments when for example, managing the lifecycle of thousands of assets from engineering to operate and maintain to retirement and the associated information flows.

This is not to say that traditional models are obsolete or unimportant, especially for applications that have a fairly static information model or complex transactional processing. Semantic models are just more suited for supporting an 'ecosystem' of federated data, especially in an environment where change is the norm.

Using our vehicle example, let’s also add that “thing1” has a kangaroo in the back – something that would not likely have been anticipated in a master schema. We simply add a statement to the existing table asserting the fact that thing1 isCarrying kangaroo.

There is no schema change required because it is just another statement. It is simple to add new repositories and new attributes to existing tables.
You can create a virtual data warehouse that spans multiple data silos, leaving the data as is. This is known as “data federation”. Traditional systems require creating a new data repository and copying all the necessary operational data to the data warehouse. This approach delays access to merged information, requires software and hardware to copy the data, and requires someone to maintain the new repository whenever a source system changes. The Intuition semantic model gets the benefits of federating data without requiring data to be copied. Intuition automatically detects new data, which significantly simplifies the job of updating the model as things change.

Illustration of a directed graph: subject and object nodes with labeled arcs showing the predicate.

Intuition uses Resource Description Framework (RDF) to describe “things” and the relationships between them as statements, in the form of: subject, predicate, object. Taken together, these RDF statements form a “graph” or ontological model. The example above shows the Intuition plant model graph federating two other ‘databases’: a real-time database (tag, value), and a maintenance system database (work orders associated with an asset).

Different graphs can share nodes or branches. This shared information provides the ‘links’ that allow the graphs to be merged temporarily.
Illustration of a Semantic Model with Aggregated Graph: Aggregation enables data federation.

Once we have the merged graph, we can create queries that span multiple data sources. The data federation is ‘temporary’ – it is merged for the duration of the query only and the master information source keeps the original data. This is a key benefit that supports the low overhead cost of semantic modeling. Federation also allows you to use the Intuition model without needing to know about the data structures and naming conventions of the other data sources.

Going back to our example, the following tables illustrate how subsets of statements can exist in isolated repositories.

**Conversation Repository**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ranchero</td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ute</td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>El Camino</td>
</tr>
<tr>
<td>Ranchero</td>
<td>owl:sameAs</td>
<td>sportyUte</td>
</tr>
<tr>
<td>El Camino</td>
<td>owl:sameAs</td>
<td>Ute</td>
</tr>
</tbody>
</table>

**Ranchero Repository**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranchero</td>
<td>wasMadeIn</td>
<td>1979</td>
</tr>
<tr>
<td>Ranchero</td>
<td>rdf:type</td>
<td>car</td>
</tr>
<tr>
<td>Ranchero</td>
<td>rdf:type</td>
<td>truck</td>
</tr>
<tr>
<td>blueThing</td>
<td>rdf:type</td>
<td>Ranchero</td>
</tr>
<tr>
<td>blueThing</td>
<td>hasManufacturer</td>
<td>ford</td>
</tr>
</tbody>
</table>
Each repository above can exist in isolation and then be stitched together at query time. This is the nature of federation.
You can embrace standards in a non-standard environment.

Data repositories often grow organically and without standards in mind. It is a costly proposition to apply standards to migrate legacy data sources. The Intuition semantic model allows us to create a layer on top of the native data sources and re-shape it into a standard model. We can then create dashboards, reports, and other visualizations that are written as if the native data was using ISA-95, ISO-15926, MIMOSA, or other standards.

For instance, let’s take the example dataset and create a new inference rule to show how we can apply a standard ontology to non-standard data.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ranchero</td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>Ute</td>
</tr>
<tr>
<td>thing1</td>
<td>rdf:type</td>
<td>El Camino</td>
</tr>
</tbody>
</table>

This inference rule says, “Every Ranchero is an ISO-15926 Arranged Individual”. ISO-15926 defines the arranged individual as “An individual that is an arrangement of components.”

\[
\begin{align*}
\text{WITH} & \nonumber \\
& \text{CONSTRUCT} \\
& \{ \\
& \quad ?\text{subject rdf:type iso15926:ARRANGED\_INDIVIDUAL} \\
& \} \\
\text{WHERE} & \nonumber \\
& \{ \\
& \quad ?\text{subject rdf:type Ranchero} \\
& \} \\
\end{align*}
\]

The following query asks the system for all “Arranged Individuals”.

\[
\begin{align*}
\text{SELECT} & \text{?subject} \\
\text{WHERE} & \{ \\
& \quad ?\text{subject rdf:type iso15926:ARRANGED\_INDIVIDUAL} \\
& \} \\
\end{align*}
\]

The following result is produced through inference because we know that thing1 is a Ranchero and therefore also an Arranged Individual. It is unnecessary to explicitly store the fact that thing1 is an arranged individual (although it can be done for efficiency). This allows us to “think” in standards even though the underlying data is stored in non-standard formats.

<table>
<thead>
<tr>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing1</td>
</tr>
</tbody>
</table>
Embracing MURA as a standards-based architecture in a non-standard environment

The Microsoft Upstream Reference Architecture (MURA) describes a set of foundational principles: performance-oriented IT infrastructure, enhanced user experience, optimized domain-specific infrastructure, rich application platform, and comprehensive interoperability. Standards-based data architectures (aligned with both Microsoft’s Open Specification and appropriate industry standards) supported by a semantic model is an efficient way to promote interoperability between disparate and incompatible data sources.

Microsoft Upstream Reference Architecture – Foundational Principles

Embracing Intuition as a standards-based architecture in a non-standard environment

Intuition was designed to facilitate knowledge management and deliver actionable information. It does this by supporting an enterprise’s federated information topology and enabling seamless access to source data to maximize interoperability, providing tools to use data intelligently, and to maximize the user experience and actionable outcomes. The Intuition architecture is closely aligned with MURA principles and component structure.

The following diagrams illustrate the natural relationships between the conceptual and component architectures of MURA and Intuition.¹

¹ MURA calls for “Semantic Data Services” to be established as the access layer to the multiple data repositories.
Microsoft Upstream Reference Architecture – Intuition Concept Overlay

**Intuition**

**Present**
SharePoint portal XAML standard GUI

**Orchestrate**
MS Workflow

**Contextualize**
OPC UA server & Semantic DB hosting multiple data model standards: OPC UA, ProdML, ISA-95, MIMOSA, PPDFMA, etc.

**Applications**
- Well Performance Monitor
- Mobile Equipment Monitor
- Wind Asset Monitor
- Matra Bro Mine to Port

**Measure**
OPC/OPC UA/Web Services/RSS
Universal Data Access
For more information on Intuition, visit: http://www.HoneywellProcess.com/DigitalPlant

For more information on the Microsoft Upstream Reference Architecture (MURA) initiative, including white papers on the vision and road map, visit http://www.microsoft.com/mura.

Honeywell Process Solutions
1250 West Sam Houston Parkway South
Houston, TX 77042

Lovelace Road, Southern Industrial Estate
Bracknell, Berkshire, England RG12 8WD

Shanghai City Centre, 100 Junyi Road
Shanghai, China 20051