Currently, WS-Choreography (WS-CDL) defines an <import> element that imports a choreography at the package level that overwrites the existing choreography. Import notes Import definitions occur at the <package> level. The semantics:

- Allow the association of other namespaces in an imported definition.
- Necessitate that all choreographies in the package share the same token type, channel and participant-role characteristics.
- Evaluate imports in the order they occur.
- Requires the import definition must be used if specified.

The import definitions should support WSDL, schema and BPEL types. The current (27 April 2004) schema fragment is:

.....
<package
name="ncname"
author="xsd:string"?
version="xsd:string"
targetNamespace="uri"
xmlns="http://www.w3.org/2004/04/ws-chor/cdl">
importDefinitions*
informationType*
token*
tokenLocator*
role*
relationship*
participant*
channelType*
Choreography-Notation*
</package>**.....**

.....
<importDefinitions>
<import namespace="uri" location="uri"/>+
</importDefinitions>.....

The assumption with <import> was it would be compatible with or support WSDL, XML and BPEL. The following issues have been raised about the <import> functionality with partial recommendations or questions to the working group:

Issue #	***Description***	***Comments***
Issue 469 Names and Namespaces	The semantics are associated to bringing in names from namespaces different that the current one. It can never replace existing names.	**Consider use of substitution.**
Issue 484: Top level import only	Import statements need to apply potentially other points within the choreography apart from the top level.	***Defer to a later version. This could have substantive impact on other CDL structures. Need to evaluate against compatibility constraints as well (with BPEL for example).***

Issue #	*Description*	*Comments*
Issue 485: Overriding	How are definitions overridden?	**Substitution** **Need more information to evaluate further.** **Are type and name sufficient to allow the import to occur? Why is import used rather than substitution?**
Issue 561: Collisions, Perform	1. How are collisions handled? 2. Are the variables, types, tokens etc. required to be consistent between the choreography and the interaction? 3. (editorial) Make it explicit the choreography depends on message exchange only. 4. Complex conditions could apply that are not held in the choreography that affect import.	**1. Resolve with Issue 484.** **2. Resolve with Issue 484.** **3. Provide editorial statement.** **4. Allow implicit reference to a semantic constraint via the extensibility element. *Evaluate if explicit reference made with Issue 484 resolution*.**
Issue 609: Rules outside of the choreography	May be importing rules outside of the choreography	**Current function allows use of the extensibility element that could hold the reference to a semantic constraint (implicit constraint).**
Issue 611: Import dependency on rules	What happens when a composed choreography has dependencies outside of itself and it is imported?	**Place a constraint on the imported choreography. Evaluate if this is handled as an implementation or tools issue, with implementer's note in the specification. *Also relates to Issue 484 resolution*.**
Issue 687: Import vs. perform	Need to clearly differentiate the two.	**Editors to clarify that a performed choreography declared outside of the package must be imported first.**

Recommendations:
Near-term capabilities:

- Consider adding a type attribute to remain consistent with recent changes in WS-BPEL.
- Consider, for example, if we have a transaction mapping, we may have other mappings such as references that place constraints on the import, perform, transactions, etc. Therefore, do we allow these seeming contexts to be handled separately or allow another explicit generic element to do so or use the extensibility element (tExtensibleElements) [implicit]?
- If the latter, recommend we re-evaluate the constraint on this extensibility element: "Extensions MUST NOT change the semantics of any element or attribute from the WS-CDL namespace." For example, the current import semantics state that the inclusion of an imported choreography overwrites what exists, including parameters. Business process constraints (i.e. business semantics) may dictate that the parameters on the choreography that the time-to-

complete is 1 day rather than the existing choreography that specifies 2 days duration (due to performance contractual responsibility).
- In the specification, indicate that an imported or performed choreography (that is imported first), must not carry its own dependencies that could affect the choreography to which it is imported.

Future capabilities:
- Consider allowing import only at the interaction level. Note, suggest this be addressed in a later version once more research is conducted. May have implications to error handling, dependencies, etc. If interaction level import is allowed, need to ensure consistency in import to variables, types and tokens at the level of the import (i.e. Schema, WSDL or BPEL). Determine if this creates a potential conflict with BPEL support, that limits import to the process level only.
- Consider what impact a semantic constraint has on a required import definition. For example, if a time-to-complete on a choreography is 2 days and the imported definition indicates 3 days, a business semantic constraint may either: (a) not allow the import to occur or (b) allow the 2-day time-to-complete to be maintained.

Editorial:
- Explicitly indicate that import is limited to message exchange view and does not include semantical constraints that may influence it although it may reference higher-level constructs that define those semantic constraints.
- Isolation dirty-write allows immediate overwriting from other choreographies. Need to specify that this applies to import and performed choreographies. If so, and an implicit or explicit reference is used (see recommendations above), then indicate dirty-write may not be allowed (constraint applies).

Schema changes for near-term recommendation capabilities only:

> **<importDefinitions>**
> **<import namespace="uri" location="uri"** ***importType="uri"*** **/>+**
> **</importDefinitions>.....**
>
> ***Note: No change for extensibility for a semantic reference if that reference is implicit. The explicit case is not modeled because not enough is understood about the impacts in doing so.***