Published in Journal of Advanced Transportation (Article ID 7905140, 2018)

Trajectory Specification Language for Air Traffic Control

Russell A. Paielli*
NASA Ames Research Center, Moffett Field, California, 94035

Abstract

Trajectory Specification is a method of specifying aircraft trajectories with tolerances such that the position at any given time in flight is constrained to a precisely defined bounding space. The bounding space at a given time is defined by tolerances relative to a reference trajectory that specifies position as a function of time. The tolerances are dynamic and are based on the aircraft navigation capabilities and the traffic situation. This paper proposes a standard Trajectory Specification Language (TSL) based on the Extensible Markup Language (XML) to represent these specifications and to communicate them by datalink. The language can be used to downlink trajectory requests from air to ground and to uplink trajectory assignments from ground to air. The XML format can be converted to binary for operational use, if necessary, using Efficient XML Interchange (EXI) or Abstract Syntax Notation (ASN.1).

I. Introduction

Air traffic control is currently performed by human controllers using radar displays of traffic and voice communication with pilots. The number of flights that a controller can reliably manage at one time, however, is substantially less than the number that could safely fly in the airspace with an automated ATC system.[1,2] Controllers are remarkably reliable overall, but they are human and therefore make mistakes. Over 1,800 operational errors (breaches of minimum required separation officially attributed to controller error) occurred in one recent year in the US, including 55 serious cases in which "a collision was barely avoided."[3] Automation can reduce human error, but an autonomous ATC system that works for all possible traffic situations and conditions is difficult to design and implement and is even more difficult to verify and validate to the required level of reliability and integrity.

Trajectory Specification is a proposed far-term enhancement of the Advanced Airspace Concept (AAC) being developed by NASA for automating ATC in both enroute airspace[4–6] and the terminal airspace around major airports.[7,8] The Trajectory Specification concept was first published in 2005[9] and has been updated in more recent publications[10–12] (and issued a US patent). A similar proposal by others[13] followed several years after the first publication on the concept.

The main idea of Trajectory Specification is to limit the allowed deviation from an assigned reference trajectory so that the aircraft position at any given time in flight is constrained to a precisely defined volume of airspace. As will be explained later in the paper, the bounding volume at any time is defined by tolerances relative to a reference position at that time as the flight advances along its route. Although near-term applications are possible, the full concept is considered "far-term" because it requires new aviation standards and a new generation of airborne Flight Management Systems (FMS).

*Aerospace Engineer, Code AFT, MS 210-10, Russ.Paielli@nasa.gov, AIAA Associate Fellow. The author declares that there is no conflict of interest regarding the publication of this paper.

Trajectory Specification generalizes Required Navigation Performance (RNP)[14,15] to the longitudinal plane by adding vertical and along-track tolerances to the cross-track tolerances that are already part of RNP. Dynamic RNP[16] allows routes to be created and assigned dynamically, and it also allows discrete altitude constraints and a required time of arrival (RTA), but it does not specify a continuously bounded trajectory.

Among the potential benefits of Trajectory Specification is the mitigation of risks that arise in the case of a system outage. Safety must be maintained even if the ATC system or the datalink goes down for an extended period of time while traffic density is too high for a human controller to safely take over and manage the traffic. One alternative is to stop any new traffic from entering the affected airspace when its ATC system or datalink fails.[17] The current traffic will then exit the affected airspace (or land as planned) within approximately 10 to 15 minutes based on the deconflicted trajectories assigned by AAC. The deviation from those trajectories is not bounded, however, so conflicts could still arise due to inaccuracies in the winds, weight, or thrust levels that were used to predict the trajectories. The problem could be mitigated by adding an extra separation buffer to the assigned trajectories, but that would diminish airspace capacity during normal operation.

By explicitly bounding deviation from the assigned trajectory in all three axes, Trajectory Specification can guarantee safe separation between flights for as long as they remain in conformance with the tolerances of their assigned trajectories, out to the conflict-free time horizon that was computed. That conflict-free time horizon would normally be on the order of 15 to 30 minutes or more, depending mainly on the current wind modeling accuracy. If the ATC system or datalink goes down, the previously deconflicted and assigned trajectories will remain active in the FMSs to keep the flights safely spaced and separated.

As a fundamentally proactive rather than reactive approach to ATC, Trajectory Specification can also provide safety benefits during normal operation. Rather than simply relying on continuous conflict detection and tactical maneuvering when necessary to correct for prediction errors, it facilitates more rigorous, precise, and predictable strategic planning. Tactical backup systems[18,19] will still be needed, but they should have to intervene less often. The airborne collision avoidance system (ACAS) would also still be maintained as an emergency backup. The added precision and predictability provided by Trajectory Specification could also facilitate closely spaced parallel approaches or, eventually, formation landing of more than two flights in closely spaced formations to increase runway landing rates.

This paper proposes a standard Trajectory Specification Language (TSL) based on the Extensible Markup Language (XML) to represent these specifications and to communicate them by air/ground datalink. The language can be used to downlink trajectory requests from air to ground and to uplink trajectory assignments from ground to air. The underlying datalink technology that would be used is outside the scope of this paper, but a likely candidate is the developing Internet Protocol Suite (IPS) for Air Traffic Services (ATS). The paper does not formally define an XML schema but rather shows how the format might be structured and what information it should contain.

Researchers at Boeing have developed the Aircraft Intent Data Language (AIDL),[20] which captures the details of how the pilot intends to fly the aircraft. This language provides detailed input for a ground-based trajectory predictor, thereby facilitating more accurate ground-based trajectory predictions for ATC. However, trajectory predictions based on AIDL depend on wind modeling, and large wind modeling errors at any particular time can cause large trajectory prediction errors. The FMS will not have the data it needs to compensate for the wind errors and bound the resulting position error. TSL, on the other hand, provides the predicted trajectory to ATC directly from the FMS and provides tolerances for the FMS to stay within bounds that can guarantee safe separation.

TSL is also similar in some respects to the Extended Predicted Profile (EPP), a downlink

capability that was recently added to Automatic Dependent surveillance - Contract (ADS-C). EPP can downlink predicted state and trajectory data for up to 128 TCPs (Trajectory Change Points: changes in target altitude, heading, or speed). However, not all variables that affect the trajectory reconstruction are included in EPP, hence predictions based on it can still have significant error, particularly during climb and descent.[21] The time between TCPs can be over 20 minutes even during climb and descent. But even if the TCPs were much closer together in time, that would not improve the accuracy of the underlying predictions, which depend on wind modeling. Like AIDL, EPP can improve trajectory prediction accuracy, but it does not bound the errors as TSL is designed to do.

TSL overlaps in scope to some extent with the Flight Information Exchange Model (FIXM),[22] a globally standardized flight data exchange model based on XML. FIXM includes some of the same trajectory data that is in TSL, but it does not fully specify a trajectory with continuous tolerances as TSL does. FIXM is designed primarily for coordination between ATC systems and facilities as well as airline operational centers. Although It will eventually be available on the flight deck, it is not designed for real-time operational coordination between ATC and aircraft in flight. Note also that, like FIXM, trajectory assignments based on TSL must be shared with all ATC facilities that will handle a particular flight.

The remainder of the paper is organized as follows. The next section outlines the Trajectory Specification concept for background. Section III presents the proposed Trajectory Specification Language. Section IV briefly discusses the data transfer requirements of the language and compares it with a common consumer data streaming application for entertainment. The paper ends with a brief summary, and an appendix briefly introduces and discusses XML, EXI, and ASN.1.

II. Trajectory Specification Concept

Trajectory Specification is essentially the construction of dynamic, virtual roadways, corridors, or tubes in the sky using data standards, an air/ground datalink, and software to specify the parameters. Because the parameters are a continuous function of time, it is more precise, more continuous, more dynamic, and more flexible than the static published routes and discrete altitude restrictions that are currently used to organize traffic and separate arrival streams from departure streams in terminal airspace.

A route is the vertical projection of a trajectory onto the surface of the earth. In the Trajectory Specification concept, a route consists of alternating straight (i.e., great circle) segments and circular turn arcs. Any point along the route can be specified by the distance along the route relative to an arbitrary reference point on the route (positive in the direction of flight), and that distance will be referred to as the along-track distance or position. A useful convention for terminal airspace is to define the runway threshold as the zero reference point, so that departure trajectories start at, and arrival trajectories end at, zero along-track distance. Then the along-track position indicates the distance flown from takeoff or yet to be flown to landing.

Figure 1 shows an example of a plan view of trajectory bounds at an instant in time. The lane width is twice the cross-track (lateral) tolerance. The along-track bounds at a point in time are vertical rectangles normal to the route direction (which appear as line segments in the plan view) and are defined by the along-track tolerances relative to the reference position at that time. The along-track bounds combine with the cross-track bounds to form a bounding area in the plan view that is a rectangle in the straight segments, an annular area in the turns, or a combination of the two, as shown in the figure.

Figure 2 shows an example of a side view of trajectory bounds at an instant in time during a climb. The along-track bounds combine with the vertical bounds to form a shape with straight vertical sides and curved top and bottom in the longitudinal plane. In level flight, the top and

bottom would also be straight. The vertical tolerances in level flight could be ±100 or ±200 ft, but in climb or descent they could be larger, on the order of ±1000 ft or more, depending on the traffic situation, because altitude is more difficult to predict and control in climb or descent. The tolerances can vary as a function of along-track distance, but the function itself will be fixed at the time of assignment (or reassignment).

The bounding volume at each point in time is a segment or "slice" of a stationary (earth-fixed) bounding tube through which the aircraft is required to fly. Each tube is dynamically constructed for one flight. The vertical cross-sections of the tube normal to the route direction are vertical rectangles, and position along the tube is temporally constrained. (These tubes should not be confused with another tube concept that allows many flights to fly in parallel in a single tube like cars on a freeway.) If one such bounding tube goes over or under another with sufficient vertical separation, then separation is guaranteed as traffic on a freeway is guaranteed to be separated from traffic on a road that goes over or under the freeway. If two such bounding tubes intersect or are separated by less than the minimum allowed separation between flights, the specifications must guarantee separation temporally

Trajectory Specification is an extension of trajectory prediction. Trajectory prediction should normally be done by the FMS, which takes the current flight state, the flight intent, and wind data as inputs and computes a trajectory prediction based on an aircraft performance model. Alternatively, an advanced Electronic Flight Bag (EFB) could potentially be used for this purpose. The intent includes the route, airspeed, cruise altitude, and possibly other parameters. The FMS can also (optionally) receive, record, and take into account the currently assigned trajectories of other flights in the vicinity to avoid conflicts while computing its own trajectory request. Taking other trajectories into account will increase the probability that the requested trajectory will be free of conflicts and approved by ATC without modification. The FMS could use some of the the same software components that implement the conflict detection and resolution to be discussed later.

The FMS (or EFB) then downlinks the predicted trajectory to ATC as a request. ATC takes the predicted trajectory as an input and adds default tolerances. It then checks the trajectory for conflicts with the current trajectory assignments of other flights and modifies it to resolve conflicts, if necessary, then uplinks it back to the FMS as the assigned trajectory. If a conflict-free trajectory cannot be found, the flight will be delayed until one can be found (by delaying takeoff time for a departure or putting an arrival into a holding pattern near the terminal airspace boundary). The pilot (or the FMS, if programmed to do so) can request a new or updated trajectory at any time, and the ATC system should approve it if there are no conflicts or constraint violations. The ATC system will generate a new or updated trajectory whenever necessary to resolve a conflict.

The basic operational concept can be summarized as follows:

- FMS records trajectory assignments from ATC for other flights (optional)
- Pilot enters route and intent data into FMS
- FMS computes a deconflicted trajectory prediction
- FMS downlinks trajectory prediction to ATC as a request
- ATC assigns tolerances, checks for conflicts and constraint violations
- ATC modifies trajectory to resolve conflicts and violations if necessary
- ATC uplinks assigned trajectory with tolerances
- FMS flies assigned trajectory to specified tolerances

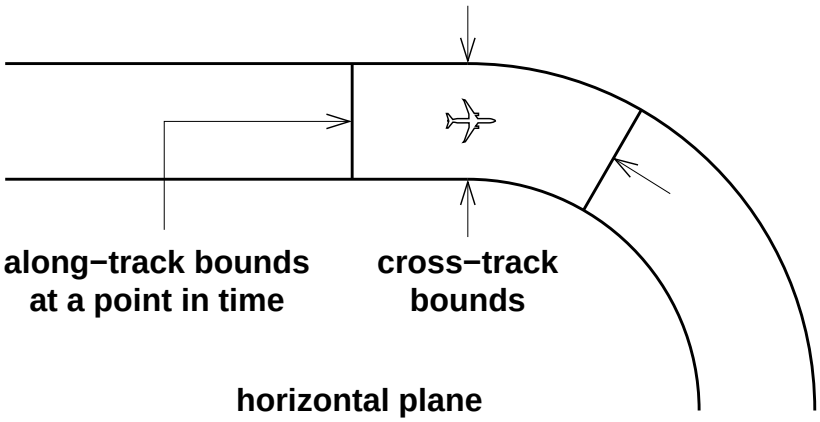

Figure 1. Trajectory bounds in the horizontal plane

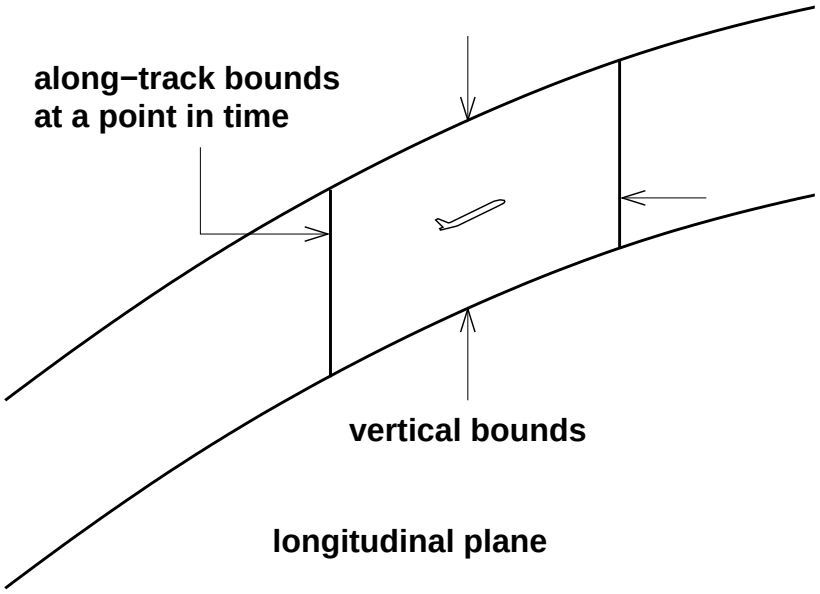

Figure 2. Trajectory bounds in the longitudinal plane

Note that the conflict checks by ground-based ATC are essential for several reasons even if the FMS is programmed to receive, and avoid conflicts with, all trajectory assignments to other flights. Firstly, the FMS could miss a trajectory assignment to another flight if the assigning ATC system is out of radio range at the time of the assignment or the signal is not accurately received for any reason. Secondly, an assignment to another flight could occur while the FMS is computing its own trajectory request, resulting in a potentially dangerous race condition. As an added benefit, the conflict checks on the ground serve as a backup in case of an error in the FMS conflict detection software.

Trajectory tolerances will depend on the aircraft navigation capabilities and the traffic situation. The navigation capabilities determine the lower limit of feasible tolerances, and the traffic situation determines the upper limit. The FMS will know its own minimum tolerances but has no inherent incentive to keep its requested tolerances to a minimum, which is why the tolerances should be provided by ATC from an established database of minimum tolerances for each aircraft or aircraft type.

In general, vertical and along-track tolerances would be made as large as reasonably possible while guaranteeing safe separation. The tolerances could be completely disabled or made arbitrarily large when they are unnecessary. The thrust and airspeed adjustments that are necessary to maintain conformance should be relatively small except in rare cases when the wind model that was used to generate the reference trajectory was grossly in error. Periodic updates can adjust for the accumulated effects of wind errors. If the tailwind is stronger than predicted, for example, the reference trajectory can be shifted in time periodically to re-center the flight, but only if the shift causes no conflict and violates no time constraint.

Trajectories in terminal airspace would normally be assigned shortly before entry into the airspace, perhaps 1 or 2 minutes before entry for arrivals, and perhaps 30 seconds before the start of takeoff roll for departures. In either case, a tentative trajectory could be computed and assigned earlier and modified at final assignment time, if necessary, to avoid conflicts. Once assigned, trajectories should normally not need to be modified for the entire 10 to 15 minutes that is typically spent in terminal airspace.

The full Trajectory Specification concept requires both a ground-based ATC component and an airborne FMS component. The focus of this paper is the ATC component, for which prototype software has been developed using functional programming methods with the Scala programming language. The resulting software is intended to form the basis for a sound software architecture as well as a starting point for an actual operational implementation. The airborne component is outside the scope of this paper, but a brief overview will put it into perspective.

The function of the airborne component is to understand the trajectory specifications and keep the flight in conformance with its assigned trajectory to within the specified tolerances. Due to safety criticality and extreme certification requirements, the airborne component will require a major development effort, but the basic ideas are not complicated. The existing flight-control feedback loop can have a lower-bandwidth outer loop wrapped around it to keep the flight within its bounds. It will monitor for proximity (and predicted proximity) to the bounds and make adjustments as necessary. Vertical speed will be adjusted to maintain vertical conformance, and airspeed will be adjusted to maintain along-track conformance. These adjustments will be similar to what a pilot could do through the Mode Control Panel (MCP), but they will be automated.

III. Trajectory Specification Language

The Trajectory Specification concept will require a standard language to represent and communicate the specifications between aircraft and the ATC system. A proposed language called the Trajectory Specification Language (TSL) has been developed based on XML and will be presented in this section. TSL can be used to downlink trajectory requests and to uplink trajectory assignments, and a conforming FMS will be programmed by the manufacturer to understand the language and to keep the flight in conformance. The objective of this paper is not to formally define an XML schema but rather to show how the format should be structured and what information it should contain. Example XML elements will be presented and explained to provide a high-level design that can be used to develop a formal Schema.

A language based on XML was proposed in the original paper on Trajectory Specification[9] in 2005. At that time, Trajectory Specification was a concept with no prototype implementation, but now a software prototype has been developed, and the proposed language has evolved substantially. A software prototype has been developed for the ground-based ATC component of the Trajectory Specification concept using functional programming style in the Scala programming language. The XML elements to be discussed below are serializations of the main classes in that software.

As a serialization of the software classes, the TSL has also been useful for software development. Fast-time simulation testing on a full day of traffic can take over an hour to run, and if a conflict is not resolved properly, the TSL can be used to capture the detailed traffic state at the time of the conflict. That capability allows the developer to restart the simulation at the desired state rather than having to wait up to an hour for the same traffic situation to develop in order to test each software change. This usage does not require a DTD or schema.

The design of TSL involves many choices that reasonable people can disagree on, particularly involving the degree of structure. Should each coordinate of a position be a subelement, for example, or should the coordinates simply be concatenated together into a comma-separated list of numbers? XML purists will argue that more structure is better because it reduces the chance of an error. For most XML applications that may be true, but it is not necessarily true for this safety-critical application because the barrier to entry is very high. FMS certification is a rigorous process with extensive testing, and errors involving the order of coordinate could never survive it. Hence, for simplicity, the approach taken in this paper is to avoid some low-level structure, but that can easily be changed if a future standards committee decides to do so.

Standard aviation units are used as shown in Table 1. Time is given in seconds (sec), horizontal distance or length is in nautical miles (nmi), vertical length or altitude is in feet (ft), and angles are in degrees (deg). Allowing alternate units can enhance flexibility but also increases the risk of error; whether or not to allow them will be deferred to a future standards committee. If alternate units are not allowed, the unit attributes shown in the examples to follow will not be needed. If alternate units are allowed, they should be from a very small set of options to minimize the risk of error because all users need to be able to recognize and convert them.

Table 1. Physical units

quantity	unit
time	second (sec)
horizontal length and position	nautical mile (nmi)
vertical length and altitude	foot (ft)
angle	degree (deg)

A. Trajectory

The specification for a trajectory is represented by the `traj` element, the structure of which is shown in XML sample 1. The `name` attribute could be the call sign or any other unique and appropriate identifier. The `time` attribute is the assignment or request time in seconds (unix time). The optional `assign` attribute is a boolean that defaults to false and is true if the trajectory is an assignment (as opposed to a request or a resolution maneuver candidate). A possible alternative to the boolean `assign` attribute could be a `status` attribute that is limited to a small set of enumerated values, such as `ASSIGN`, `REQUEST`, and `CANDIDATE`. The `GUFI` attribute holds the Globally Unique Flight Identifier, which is required for all ATC flight data transactions.

The subelements of the `traj` element are `flight` (flight information), `route`, `refTraj` (reference trajectory), `altTols` (altitude tolerances), and `alongTols` (along-track tolerances). As explained earlier, these elements correspond to the Scala classes that were developed to implement the concept. Although not shown, each subelement could have its own optional `name` attribute for reference (which should all be the same).

```
<traj name="AAL123" time="1378478953" assign="true" GUFI="...">
    <flight name="AAL123" info="B738 Large DFW 18R Arr"/>
    <route> ... </route>
    <refTraj> ... </refTraj>
    <altTols> ... </altTols>
    <alongTols> ... </alongTols>
    </traj>
```

XML Sample 1. Trajectory structure

The `flight` subelement contains basic flight information in its `info` attribute, including the aircraft type code (B738 in this example), the weight class (Large), the airport code (DFW), the runway (18R), and the flight type (Arr for arrival or Dep for departure). This information could be broken down into individual subelements as shown in XML sample 2, which is more explicit and can be parsed using standard XML tools. However, the simpler form reads naturally and is trivial to parse. A standard equipment code could be appended to the aircraft type code if necessary (after a slash, as usual). The other subelements of the trajectory (`traj`) element are explained in the following subsections.

B. Route

The route is the planview of the trajectory, consisting of straight segments and turn segments. The structure of the `route` element is shown in XML sample 3. The basic flight information (`flight`) from the `traj` element is repeated for reference because the route depends on that information. The reference along-track starting distance (`startDist`) is arbitrary and was set to -53.031923 nmi to make the along-track distance zero at the end of the trajectory (at the runway threshold). The cross-track tolerance (`crossTol`) is set to 0.6 nmi, which is equivalent to RNP 0.3.

```
<flight name="AAL123">
    <aircraft>B738</aircraft>
    <weightClass>Large</weightClass>
    <airport>DFW</airport>
    <runway>18R</runway>
    <type>Arrival</type>
    </flight>
```

XML Sample 2. Flight information

```
<route name="1366">
    <flight name="1366" info="B738 Large DFW 18R Arr"/>
    <startDist unit="nmi">-53.031923</startDist>
    <crossTol unit="nmi">0.6</crossTol>
    <waypts type="local" frame="D10" unit="nmi">
        <waypt name="HOWDY"> 27.917174, -31.050092 <rad>2.000</rad></waypt>
        <waypt> 27.235636, -27.108529 <rad>3.000</rad></waypt>
        ...
        </waypts>
    </route>
```

XML Sample 3. Route element

The waypoint list (`waypts`) of the `route` element has a `type` attribute, which can be `local` or `global`, depending on whether the waypoint locations are given in terms of a locally level map projection or global geodetic coordinates (latitude and longitude). If a local map projection is used, the `frame` attribute specifies the name of the frame that is used, which is D10 in this case, a predefined local frame for the D10 TRACON. A predefined frame (defined by its projection type and tangency point) would be published for each major TRACON and would never change. If a global frame is used, it would default to WGS-84 or whatever a standards committee deems appropriate. As explained earlier, the unit attributes could be optional and default to `nmi` if not given.

The cross-track tolerance should rarely change during a flight except possibly when entering or exiting terminal airspace. The `crossTol` subelement of the `route` element could represent such a change by appending data to the first cross-track tolerance as shown in XML sample 4. In this example, the initial cross-track tolerance of 0.6 nmi is followed after a slash by `23.5: 2.0`, which means that the cross-track tolerance changes to 2.0 nmi at the along-track distance of 23.574 nmi. Changes in cross-track tolerances are instantaneous (discontinuous) at the change point (rather than piecewise linear as for the vertical and along-track tolerances to be discussed later). Additional changes further along the route could follow similarly if necessary. Additional structure could be added here but, as explained earlier, the FMS certification process is so rigorous that basic parsing errors would not survive it.

```
<crossTol unit="nmi">0.6 / 23.5: 2.0</crossTol>
```

XML Sample 4. Cross-track tolerance element

The individual waypoints are each specified by the `waypt` element. Each waypoint can have an optional `name` attribute, which is provided for reference only and has no effect on the resulting route. The first waypoint in the example is called "`HOWDY`," and the coordinates of its position are 27.917174 nmi and -31.050092 nmi. If geodetic coordinates are used, the coordinates would be latitude and longitude in degrees. The `rad` subelement represents the turn radius at the waypoint, which is 2.0 nmi for the first waypoint in this case. The turn radius is required for all but the first waypoint for departures or the last waypoint for arrivals (any value provided for those endpoints will simply be ignored as meaningless). All turns are tangent arc "flyby" turns, and if a specified turn radius cannot be accommodated for the given the leg lengths, the route will be rejected as geometrically invalid. The use of six digits after the decimal place provides a high resolution that is nearly equivalent to binary. More digits would be needed for geodetic coordinates to achieve the same level of resolution. The ellipses indicate that several waypoints have been cut for brevity in this example.

C. Reference Trajectory

The reference trajectory is the ideal predicted trajectory that would be flown in the absence of any wind modeling errors or other modeling errors. It is represented by the `refTraj` element, the structure of which is shown in XML sample 5. The time step (`dt`) in this example is 5.0 sec, and the reference time (`refTime`) is 1378478942.2 sec. The reference time is simply the starting time of the reference trajectory and is used to save space and enhance readability.

```
<refTraj name="1366">
    <dt unit="sec">5.000</dt>
    <refTime unit="sec">1378478942.200</refTime>
    <points type="local" frame="D10" units="sec,nmi,ft">
        <pt>0.000,27.905328,-30.981547,9986</pt>
        <pt>5.000,27.837196,-30.587082,9982</pt>
        <pt>10.000,27.768812,-30.192529,9981</pt>
        ...
        <pt>750.000,7.024273,-2.354429,481</pt>
        </points>
    </refTraj>
```

XML Sample 5. Reference trajectory element

Like the `waypts` subelement of the `route` element discussed above, the points list (`points`) has a `type` attribute, which can be `local` or `global`, depending on whether the points are given in terms of a locally level map projection or global geodetic coordinates. If a local map projection is used, the `frame` attribute specifies the name of the frame that is used, which is again `D10` in this case. The `units` attributes is shown as `''sec,nmi,ft''`, meaning that the time is given in seconds, horizontal position is given in nautical miles, and altitude is given in feet.

Each point (`pt`) in the list of points has a time stamp (relative to the reference time), x and y coordinates of position, and an altitude, all separated by commas and (optional) spaces. The first point has a time of 0.0 sec, x and y coordinates of 27.905328 nmi and -30.981547 nmi, and an altitude of 9986 ft. Again, additional subelement structure could be added here, but the FMS certification process is so rigorous that basic parsing errors would not survive it. As before, the use of six digits after the decimal place provides a high resolution almost equivalent to binary. Again, if geodetic coordinates are used, the horizontal coordinates would be latitude and longitude in degrees, and more digits will be required for the same level of resolution. The ellipses indicate that many points have been cut for brevity in this example.

An alternate form of the point (`pt`) element is also possible. Rather than specifying position in terms of locally level or geodetic coordinates, it could be specified in terms of along-track distance. In that case, each point would contain the time stamp, the along-track distance, and the altitude. This alternate form has two advantages. Firstly, it reduces the number of position coordinates from two to one, significantly reducing storage space and transmission bandwidth requirements. Secondly, it eliminates the possibility that the specified point could be off the route (because the cross-track coordinate would always be zero by definition). The disadvantage, perhaps insignificant, is that the actual position on the surface of the earth is not obvious, and the potential for error may be slightly higher.

Note that the points of the reference trajectory need not be uniformly spaced in time as shown in the example. They could be spaced further apart in steady-state flight, for example, but they they should be spaced 5 seconds or less in turns and altitude transients. They will be converted internally by the ATC or FMS software to a uniformly spaced sequence of time steps specified by the `dt` element. (Uniform time steps are a key to efficient computation because they allow fast access of trajectory data points as a function of time by array indexing and interpolation). The reference trajectory typically takes up approximately 80-90% of the overall storage space for the trajectory specification.

D. Altitude Tolerances

Figure 3 shows an example of a side view of the stationary, rectangular tube in which an aircraft is required to fly. The Trajectory Specification software automatically detects the level segments and applies the default tolerance of ±200 ft in those segments. It also applies the tapered transitions between the level and non-level segments and cuts off the altitude overshoots that would otherwise precede or follow a level segment. The tapered transitions are shown at a slope of 2.5 deg but could be varied slightly. The smaller the taper angle is, the more airspace is reserved. The taper angle should be slightly less than the climb or descent angle to allow enough room for a normal leveloff or start of climb or descent.

Note that the end of climb and start of descent are clearly bounded. Lack of such bounds causes significant uncertainty for automated conflict detection, diminishing airspace capacity.[23] Discretionary descents in particular (in which the pilot is given discretion as to when to start descent) have caused problems for automated conflict detection, but they can be accurately represented, if necessary, by using larger altitude tolerances.

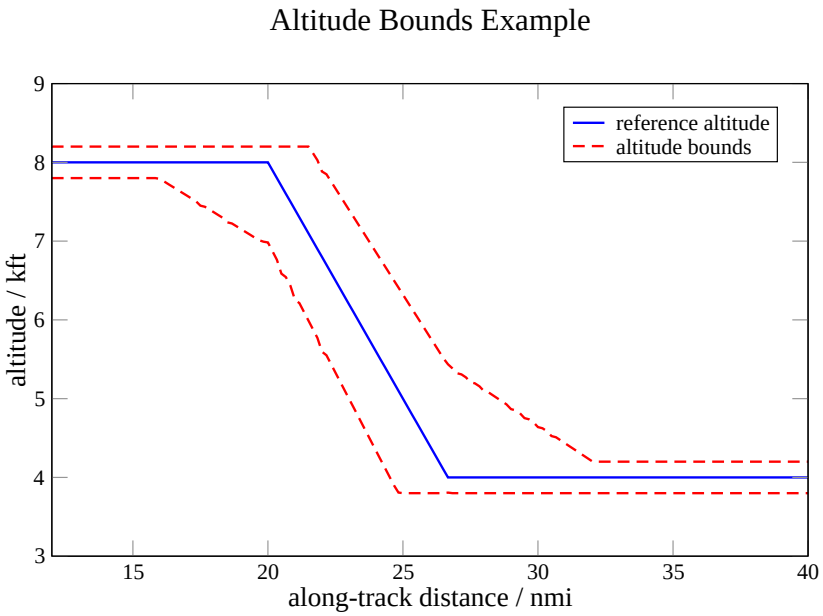

Figure 3. Simplified example of altitude bounds as a function of distance along route[12]

The tolerances for level flight are not explicitly specified but are assumed to be consistent with the current "altitude rounding" rule: if an aircraft is in level flight within $pm200$ ft of its cleared altitude, it is considered to be at its cleared altitude for purposes of separation requirements (without this rule, many nuisance alerts would occur due to small altitude deviations in cruise). The taper angle for transition to and from level flight is specified in the optional `taper` subelement, and the default value is 2.5 deg if not specified. The tolerances for level flight (including the tapered transitions to and from level flight) override the explicitly specified tolerances discussed below, which apply in climb and descent.

The altitude tolerances are specified in the `altTols` element, the structure of which is shown in XML sample 6. The piecewise linear tolerances are specified as a list of tolerance (`tol`) points, where each point contains the tolerance values separated by commas and preceded by the along-

track distance at which it applies followed by a colon (spaces are optional). The first `tol` subelement shows vertical tolerances of ±500 ft at an along-track distance of 0 nmi. The lower tolerance is given first as a negative value, followed by the upper tolerance. By definition, those tolerances apply for any along-track distance less than the first specified distance of 0 nmi. The second `tol` point shows that the tolerances increase linearly to ±1000 ft at an along-track distance of 40 nmi and, because that is the last `tol` point, remain constant beyond that distance by definition. (If there is only one `tol` point, then the tolerances are constant, and the along-track distance for that point is irrelevant.) Although the lower and upper tolerances are equal in this example, in general they can be different.

```
<altTols units="nmi, ft">
    <tol> 0: -500, 500 </tol>
    <tol> 40: -1000, 1000 </tol>
    <taper unit="deg">2.5</taper>
    </altTols>
```

XML Sample 6. Altitude tolerances element

E. Along-Track Tolerances

The along-track tolerances are specified in the `alongTols` element, the structure of which is shown in XML sample 7. Again, the piecewise linear tolerances are specified as a list of tolerance (`tol`) points, where each point contains the tolerance values separated by commas and preceded by the along-track distance at which it applies followed by a colon (spaces are optional). The first `tol` subelement below shows along-track tolerances of ±1 nmi at an along-track distance of -53.031923 nmi. The back tolerance is given first as a negative value, followed by the front tolerance.

```
<alongTols, unit="nmi">
    <tol> -53.0319: -1.0, 1.0 </tol>
    <tol> -10: -0.2, 0.2 </tol>
    </alongTols>
```

XML Sample 7. Along-Track tolerances element

The starting distance was chosen in this example to be same as the `startDist` of the route, which was set to make the along-track distance zero at the runway threshold. By definition, those tolerances apply for any along-track distance less than that first specified distance. The second `tol` point shows that the tolerances decrease linearly to ±0.2 nmi at an along-track distance of -10 nmi and, because that is the last `tol` point, remain constant beyond that distance by definition. (Again, if there is only one `tol` point, then the tolerances are constant, and the along-track distance for that point is irrelevant.) Such a decreasing along-track tolerance would be typical during an arrival

rush when throughput is critical. Again, although the front and back tolerances are equal in this example, they can be different.

The trajectory tolerances are not required in a downlinked trajectory request from an aircraft because they will be assigned by ATC based on published aircraft navigational capabilities and the current traffic situation. Allowing the pilot or FMS to specify arbitrary tolerances would not make sense because the only incentive would be to request the largest possible tolerances. If tolerances are allowed in the downlinked trajectory request, they should be the tightest tolerances that the aircraft has been determined to be capable of conforming to. A database should be established so that the ATC system can lookup the navigational capabilities of each aircraft based on the aircraft type and equipage code.

F. Trajectory Updates

Trajectory updates can often be done more efficiently without sending an entire new trajectory. A simple time shift to adjust for accumulated wind errors, for example, can be specified by the time shift rather than an entire new trajectory. An example of how that could be specified is shown in XML sample 8. As before, the time attribute represents the assignment (update) time, and the timeshift element is the time by which the reference trajectory is shifted, which is -8 sec in this case. When received by the aircraft, the FMS would generate an entire new trajectory representation for its own use, but that trajectory need not take up communication bandwidth.

```
<traj name="1366" time="1378478953" assign="true" GUFI="...">
  <timeshift unit="sec">-8.000</timeshift>
  </traj>
```

XML Sample 8. Time shifting example

As another example of a trajectory update, consider changing the tolerances without changing the route or the reference trajectory. That could be done by using the `traj` element discussed earlier and simply omitting the route and the reference trajectory if they are unchanged.

IV. Data Transfer Requirements

As mentioned earlier, the underlying datalink technology that would be used for TSL is outside the scope of this paper, but a likely candidate is the developing Internet Protocol Suite (IPS) for Air Traffic Services. This new aeronautical datalink technology is expected to significantly increase the bandwidth available for applications such as TSL.

To quantify the data transfer requirements of the proposed language, a typical terminal trajectory was selected at random and serialized using the language. Trajectories through the terminal area typically range from 10 to 15 minutes in length, and the selected trajectory was approximately 15 minutes long. The serialization was found to contain approximately 10.1 kilobytes before compression and 2.6 kilobytes after compression with gzip, for a compression ratio of approximately 3.9. (EXI is better than gzip for compressing XML but was not used here because it is not yet widely available.) The reference trajectory takes up approximately 84% of the storage space.

Enroute trajectories are usually longer than terminal trajectories, up to several hours in length, so the data quantities could be roughly an order of magnitude larger on average. However, enroute trajectories also tend to have more and longer periods of steady-state (straight and level) flight. Steady-state segments can be accurately represented with longer time steps of perhaps 30 sec or more (rather than 5 sec or less as needed for turns and other non-steady segments), reducing the amount of data required. Note also that enroute trajectories will be updated occasionally to adjust for the accumulated effect of wind errors and to resolve conflicts as they arise within the conflict-free time horizon of approximately 20 to 30 minutes. These updates should be needed perhaps once every 10 to 20 minutes or so, but many of them should be simple time shifts.

To put these data quantities into perspective, consider the amount of data involved in streaming or downloading music to a mobile device. A song of 4 minutes in length at a typical resolution of 32 kbps requires roughly 1 MB of data to be transmitted. That is enough for roughly 100 uncompressed terminal area trajectories or 400 such trajectories after compression. Those numbers should be reduced by roughly an order of magnitude for longer enroute trajectories. If these data rates are not acceptable, or if the XML processing is too slow, a binary equivalent of the proposed XML format based on EXI or ASN.1 can be used as discussed earlier.

V. Summary

Trajectory Specification is a proposed far-term enhancement of the Advanced Airspace Concept (AAC) being developed by NASA for automating ATC in both enroute airspace and the terminal airspace around major airports. The main idea is to limit the allowed deviation from an assigned reference trajectory so that the aircraft position at any time instant in flight is constrained to a precisely defined volume of airspace. Trajectory Specification generalizes Required Navigation Performance (RNP) to the longitudinal plane by adding vertical and along-track tolerances to the cross-track tolerances that are already part of RNP.

A Trajectory Specification Language (TSL) based on XML has been developed to serialize Trajectory Specifications and communicate them by datalink. The language can be used to downlink trajectory requests from air to ground and to uplink trajectory assignments from ground to air. The proposed language can serve as a starting point for the development of a communication standard for the Trajectory Specification concept. The language is flexible and directly readable by humans, and the bandwidth requirements are modest compared to common consumer data streaming and downloading applications. If necessary, standard compression or efficient binary forms based on EXI or ASN.1 can be used.

References

[1]Mercer, J.; Homola, J.R.; Cabrall, C.D.; Martin, L.H.; Morey, S.E.; Gomez, A.N.; Prevot, T.: "Human-Automation Cooperation for Separation Assurance in Future NextGen Environments," *Proceedings of the International Conf. on Human-Computer Interaction in Aerospace (HCI-Aero 2014)*, Santa Clara, CA, 2014.

[2]Prevot, T.; Homola, J.R.; Martin, L.H.; Mercer, J.; Cabrall, C.D.: "Toward Automated Air Traffic Control — Investigating a Fundamental Paradigm Shift in Human/Systems Interaction," International J. of Human-Computer Interaction, 28:2, 77-98, Special Issue on NextGen, 2012.

[3]Guzzetti, J.B.: "FAAs Progress and Challenges in Advancing Safety Oversight Initiatives," US Dept. of Transportation, April 16, 2013.

[4]Erzberger, H.: "Automated Conflict Resolution for Air Traffic Control," 25th International Congress of the Aeronautical Sciences (ICAS), 2005.

[5]Erzberger, H.; Paielli, R.A.: "Concept for Next Generation Air Traffic Control System," *Air Traffic Control Quarterly*, Vol. 10(4)(2002), pp 355-378.

[6]Erzberger, H.; Lauderdale, T.A.; Chu, Y.C.: "Automated Conflict Resolution, Arrival Management,

and Weather Avoidance for Air Traffic Management," *J. Aerospace Engineering*, 2011 (full ref)?

[7]Nikoleris, T.; Erzberger, H.; Paielli, R.A.; Chu, Y.C.: "Autonomous System for Air Traffic Control in Terminal Airspace," *AIAA Aviation Technology, Integration, and Operations (ATIO)* Conf., Atlanta GA, 16-20 June 2014.

[8]Erzberger, H.; Nikoleris, T.; Paielli, R.A.; Chu, Y.C.: "Algorithms for Control of Arrival and Departure Traffic in Terminal Airspace," *Journal of Aerospace Engineering*, DOI: 10.1177/0954410016629499, Feb 2016.

[9]Paielli, R.A.: "Trajectory Specification for High-Capacity Air Traffic Control," *AIAA Journal of Aerospace Computation, Information, and Communication*, vol. 2, no. 9, Sept 2005.

[10]Paielli, R.A.: "Trajectory Specification for Automation of Terminal Air Traffic Control," AIAA Guidance, Navigation, and Control Conference, AIAA SciTech Forum, (AIAA 2016-1868) https://doi.org/10.2514/6.2016-1868

[11]Paielli, R.A.: "Trajectory Specification for Terminal Air Traffic Control: Arrival Spacing," *AIAA Journal of Aerospace Information Systems*, vol. 13, no. 10, Oct 2016.

[12]Paielli, R.A.: "Trajectory Specification for Terminal Air Traffic Control: Conflict Detection and Resolution," to be submitted to ATIO and *AIAA Journal of Air Transportation*.

[13]Finkelsztein, D.M.; Sturdy, J.L.; Alaverdi, O.; Hochwarth, J.K.: "4D Dynamic Required Navigation Performance, Final Report" NASA/CR2011-217051, Feb. 2011.

[14]RTCA DO-283A: "Minimum Operational Performance Standards for Required Navigation Performance for Area Navigation," Oct 2003.

[15]RTCA DO-236C: "Minimum Aviation System Performance Standards: Required Navigation Performance for Area Navigation," June 2013.

[16]Federal Aviation Administration: "Dynamic Required Navigation Performance: Preliminary Concept of Operations," Version 1.0, RTCA Paper No. 069-14/PMC-1199, March 2014.

[17]Andrews, J.W.; Erzberger, H.; Welch, J.D.: "Safety Analysis for Advanced Separation Concepts," *Air Traffic Control Quarterly*, vol. 14, no. 1, 2006.

[18]Paielli, R.A.: "Evaluation of Tactical Conflict Resolution Algorithms for Enroute Airspace," *AIAA Journal of Aircraft*, vol. 48, no. 1, Jan-Feb 2011.

[19]Tang, H.; Robinson, J.E.; Denery, D.G.: "Tactical Conflict Detection in Terminal Airspace," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 2, 2011, pp 403-413.

[20]Javier Lopez-Leones, Miguel A. Vilaplana, Eduardo Gallo, Francisco A. Navarro, Carlos Querejeta, "The Aircraft Intent Description Language: A key enabler for air-ground synchronization in Trajectory-Based Operations," *Digital Avionics Systems Conference*, 2007.

[21]Bronsvoort, J.; McDonald, G.; Paglione, M.; Young, C.M.; Boucquey, J.; Hochwarth, J.K.; Gallo, E: "Real-Time Trajectory Predictor Calibration through Extended Projected Profile Down-Link," *Eleventh USA/Europe Air Traffic Management R&D Seminar* (ATM2015).

[22]"Flight Information Exchange Model Operational Data Description," FIXM version 4.0.0, October 31, 2016. Available from https://www.fixm.aero/

[23]Cone, A.C.; Bowe, A.R.; Lauderdale, T.A.: "Robust Conflict Detection and Resolution around Top of Descent," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conf.*, Indianapolis, IN, 17-19 Sept. 2012.

[24]Abstract Syntax Notation: http://www.itu.int/en/ITU-T/asn1

[25]Extended XML Interchange: https://www.w3.org/TR/exi/

A. Extensible Markup Language (XML)

Aeronautical datalink formats have traditionally been binary, but text-based formats such as XML and JSON (JavaScript Object Notation) are viable options for this concept. They are well established standards, and the major programming languages have libraries to parse and process them. Text is more flexible than binary data and is directly readable by humans. Text requires more storage space and bandwidth than binary data, but new technologies are available to convert to an equivalent binary form for more efficient transmission and processing, if necessary. XML is slightly more verbose than JSON, but it was chosen for this safety-critical application because it is a more established standard with more capabilities and better support.

The required structure and form of an XML document can be formally described by a Document Type Definition (DTD) or an XML schema. Alternatively, the equivalent of an XML schema can be created using Abstract Syntax Notation (ASN.1),[24] an industry standard for defining platform-independent binary data formats. ASN.1 includes XML Encoding Rules (XER) to facilitate equivalent binary and XML data formats and conversion between the two. If an ASN.1 schema is developed for the Trajectory Specification concept, both a binary and an XML version of TSL will be available. Another alternative is the Efficient XML Interchange (EXI)[25] format, an efficient binary form of XML.

An XML document consists of a hierarchy of elements, each of which can contain subelements and/or attributes. Consider, for example, the following XML element:

```
<note from="building manager" to="occupants">
    <subject>electrical maintenance</subject>
    <body>Power in building 210 will be out ...</body>
    </note>
```

The main delimiters are angle brackets, which enclose the opening and closing tags of each element or subelement. The example shows an element called `note`, which has attributes `from` and `to`, and which has subelements `subject` and `body`. Attributes are specified in the opening tag of an element, and the closing tag contains the element name preceded by a forward slash. Attribute values should be in quotes as shown. (The convention in this paper is to indent the closing tag one level more than the opening tag to better preserve the logical structure.)

Attributes are supposed to contain metadata, but the distinction between data and metadata is not always obvious. A DTD or Schema can restrict allowed values to a specified discrete set. It can also restrict attributes and elements to specified data types, such as text, boolean, integer, or decimal number. Other structural and ordering restrictions can also be imposed, but they will not be discussed here.